НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО»
ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ
Кафедра автоматизованих систем обробки інформації та управління

УДК 519.854.2

«До захисту допущено»
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

Дипломний проект

на здобуття ступеня бакалавра

з напряму підготовки *6.050101 «Комп’ютерні науки»*

на тему: «*Інформаційна система підтримки навчальної діяльності студента*»

Виконав: студент *4* курсу, групи *ІС-52*

Бобер Євгеній Олександрович _____
(прізвище, ім’я, по батькові) (підпис)

Керівник *ст.викл. Клименко О.М.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант з графічної документації *ст.викл. Халус О. А.* _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Рецензент _____ _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Засвідчую, що у цьому дипломному проекті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____
(підпис)

Київ – 2019 року

Національний технічний університет України
“Київський політехнічний інститут імені Ігоря Сікорського”

Факультет (інститут) *інформатики та обчислювальної техніки*
(повна назва)

Кафедра *автоматизованих систем обробки інформації та управління*
(повна назва)

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки (програма професійного спрямування) *6.050101*

«Комп’ютерні науки» («Інформаційні управляючі системи та технології»)

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“___”_____2019 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ СТУДЕНТУ

Бобру Євгенію Олександровичу
(прізвище, ім’я, по батькові)

1. Тема проекту « *Інформаційна система підтримки навчальної діяльності студента »*

керівник проекту *Клименко Олена Миколаївна, ст. викл.*
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом по університету від “*23*”*квітня* 2019 р. №1181-с

2. Термін подання студентом проекту “*03*”*червня* 2019 року

3. Вихідні дані до проекту

Технічне завдання

4. Зміст пояснювальної записки

1. Загальні положення: основні визначення та терміни, опис предметного середовища, огляд аналогів, постановка задачі

2. Інформаційне забезпечення: вхідні дані, вихідні дані, опис структури бази даних

3. Математичне забезпечення: змістовна та математична постановки задачі, опис та приклад виконання алгоритму

4. Програмне та технічне забезпечення: засоби розробки, вимоги до технічного забезпечення, архітектура програмного забезпечення, побудова звітів

5. Технологічний розділ: керівництво користувача, методика випробувань

5. Перелік графічного матеріалу

1. Схема структурна діяльності користувача

2. Схема структурна варіантів використання

3. Схема бази даних

4. Схема структурна класів

5. Схема структурна послідовності

6. Схема структурна компонентів

6. Консультанти розділів проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання *«02» квітня 2019 року*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів виконання дипломного проекту	Термін виконання етапів проекту	Примітка
1.	*Вивчення рекомендованої літератури*	*16.02.2019*	
2.	*Аналіз існуючих методів розв'язання задачі*	*23.02.2019*	
3.	*Постановка та формалізація задачі*	*01.03.2019*	
4.	*Розробка інформаційного забезпечення*	*11.03.2019*	
5.	*Алгоритмізація задачі*	*17.03.2019*	
6.	*Обґрунтування використовуваних технічних засобів*	*23.03.2019*	
7.	*Розробка програмного забезпечення*	*28.03.2019*	
8.	*Налагодження програми*	*03.04.2019*	
9.	*Виконання графічних документів*	*11.04.2019*	
10.	*Оформлення пояснювальної записки*	*23.04.2019*	
11.	*Подання ДП на попередній захист*	*30.05.2019*	
12.	*Подання ДП на основний захист*	*03.06.2019*	
13.	*Подання ДП рецензенту*	*05.06.2019*	

Студент _____ Є.О. Бобер
(підпис)

Керівник проекту _____ О.М. Клименко
(підпис)

№ з/п	Формат	Позначення	Найменування	Кількість листів	Примітка
			Документи текстові		
1	А4	ДП ІС-5202.1181-с.ПЗ	Пояснювальна записка	70	
2	А4	ДП ІС-5202.1181-с.ТЗ	Технічне завдання	10	
3	А4	ДП ІС-5202.1181-с.ПМВ	Програма та методика випробувань	10	
			Матеріали графічні		
4	А3	ДП ІС-5202.1181-с.ССД	Схема структурна діяльності	1	
5	А3	ДП ІС-5202.1181-с.ССВ	Схема структурна варіантів	1	
			використання		
6	А3	ДП ІС-5202.1181-с.СБД	Схема бази даних	1	
7	А3	ДП ІС-5202.1181-с.ССК	Схема структурна класів	1	
8	А3	ДП ІС-5202.1181-с.ССП	Схема структурна послідовності	1	
9	А3	ДП ІС-5202.1181-с.ССК	Схема структурна компонентів	1	

Зм.	Арк.	ПІП	Підп.	Дата	ДП ІС-5202.1181-с.ВДП		
Розробн.		Бобер Є.О..			Відомість дипломного проекту	Літ.	Лист 1 / Листів 1
Керівн.		Клименко О.М..					
Консульт.						КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52	
Н/контр.		Халус О.А.					
В.о.зав.каф.		Павлов О.А.					

Пояснювальна записка
до дипломного проекту

на тему: Інформаційна система підтримки навчальної діяльності студента

Київ – 2019 року

АНОТАЦІЯ

Структура та обсяг роботи. Пояснювальна записка дипломного проекту складається з шести розділів, містить 102 сторінки, 32 рисунків, 11 таблиць, 1 додатка, 15 джерел.

У дипломному проекті реалізована інформаційна система підтримки навчальної діяльності студента.

У розділі загальних положень були описані процеси діяльності, варіанти використання, оглянуто та проаналізовано існуючі аналоги, поставлені задача із визначеними цілями та мета дипломного проекту.

У розділі з інформаційного забезпечення були визначені вхідні та вихідні дані , описана схема бази даних.

У розділі з математичного забезпечення наведені змістовна та математична постановки задачі. Обгрунтовано метод розв’язання, а також наведено опис та приклад використання алгоритму розв’язку задачі.

У розділі з програмного забезпечення наведено структурні схеми класів, послідовності та розгортання програми.

У технологічному розділі описана інструкція користувача та проведене тестування програмного продукту.

СТУДЕНТ, ПЛАНУВАННЯ, ІНФОРМАЦІЙНА СИСТЕМА, ТЕОРІЯ РОЗКЛАДІВ, ПЕРЕРОЗПОДІЛ, ДОСЛІДЖЕННЯ ОПЕРАЦІЙ

ABSTRACT

Structure and scope of work. The explanatory note of the graduation project consists of six sections, containing 102 pages, 32 pictures, 11 tables, 1 application, 15 sources.

In the diploma project implemented an information system for supporting student`s educational activities.

In the section of the general provisions describes the processes of activity, use-case, reviewed and analyzed existing analogs, set tasks with the goals and objectives of the diploma project.

In the information support section input and output data were defined, the database schema was described.

In the section of mathematical support presents the general and mathematical formulation of the problem. The method of the decision is substantiated. The description and example of the use of the solving algorithm are described.

In the section of the software presents the structural diagrams of classes, sequencing and components of the program.

The technology section provides user guides and software test methods.

STUDENT, PLANNING, INFORMATION SYSTEM, THEORY OF SCHEDULES, REDISTRIBUTION, INVESTIGATION OF OPERATIONS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

Windows Presentation Foundation (WPF) - це система для побудови клієнтських застосунків Windows з візуальними привабливими можливостями взаємодії з користувачем, графічна підсистема в складі .Net Framework.

Model-View-ViewModel (MVVM) - це архітектурний патерн, що відділяє логіку застосунку від візуального уявлення. Складається з трьох компонентів: моделі (Model), моделі представлення (ViewModel) та представлення (View).

eXtensible Application Markup Language (XAML) – являє собою мову декларативного опису інтерфейсу, основаного на XML (eXtensibleMarkup Language), в якій реалізовано модель розділу коду та дизайну, що дозволяє програмісту та дизайнеру кооперуватися.

ADO.NET Entity Framework (EF) - являється продовженням технології Microsoft ActiveX Data (інтерфейсу програмування застосувань для доступу до даних, розроблений компанією Microsoft) та надає можливість роботи з базами даних через об’єктно-орієнтований код C#.

Microsoft SQL Server – система керування реляційними базами даних (СКРБД), розроблена компанією Microsoft. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства. Конкурує з іншими СКРБД у цьому сегменті ринку.

JavaScript Object Notation (JSON) - це текстовий формат обміну даними між комп'ютерами. JSON базується на тексті, може бути прочитаним людиною. Формат дозволяє описувати об'єкти та інші структури даних. Цей формат головним чином використовується для передачі структурованої інформації через мережу.

Теорія розкладів – розділ дискретної математики, що займається проблемами впорядкування.

Дослідження операцій — дисципліна, що займається розробкою й застосуванням методів знаходження оптимальних рішень на основі математичного моделювання у різних областях людської діяльності.

Множина – сукупність певних об’єктів довільної природи.

Елементи множини – об’єкти, що складають множину.

Цільова функція – функція, що зв’язує мету (змінну, що оптимізується) з керованими змінами в задачі оптимізації. В широкому сенсі це математичний вираз деякого критерію якості одного об’єкту (рішення, процесу) в порівнянні з іншим.

Реляційна база даних – база даних, заснована на реляційній моделі даних. Для роботи з реляційними БД застосовують реляційні СКБД. Інакше кажучи, реляційна база даних – це база даних, яка сприймається користувачем як набір нормалізованих відношень різного ступеня.

ВСТУП

Підтримка навчальної діяльності та шляхи її покращення - це завдання, що повинен ставити перед собою не тільки вищий навчальний заклад. Адже, сам студент може зробити внесок у систематизацію та спрощення даного процесу.

Важливий фактор, що впливає на навчальну діяльність, це насамперед інформаційна забезпеченість. Проте, на практиці з цим часто виникають проблеми. Пошук того чи іншого інформаційного забезпечення може викликати труднощі для студента. На даний момент, будь який електронний пристрій, нехай то телефон, планшет чи персональний комп’ютер, має в собі різного роду планувальні застосунки, засоби для збереження та накопичення інформації, проте, порізну вони не завжди достатньо ефективні, крім того в більшості випадків вони не налаштовані під конкретно навчальну діяльність.

Важливу роль грає ступінь задіяності студента до системи. Якщо студент використовуватиме застосунок, що матиме вищеперечислений функціонал, постійно - це в значній мірі покращить ефективність навчання студента. Більш того, хто, як не сам студент знає всі дрібниці навчального процесу, проблеми та актуальність різного роду навчальних документів.

Накопичення інформація за час діяльності кожного користувача несе цінність, насамперед студенту-користувачу, для пошуку найрізноманітнішої інформації, накопиченої за минулий час навчальної та спрощення процесу навчання в поточний час.

Практичне значення одержаних результатів. Розроблена система підтримки навчальної діяльності студента.

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Опис предметного середовища

На сьогодні існує велика кількість систем для підтримки навчального процесу, цілями яких є автоматизація та розвиток процесів обміну інформації, її збору, обробки, збереження та представлення. В основному, такі системи призначені для різних учасників навчального процесу, як наприклад, викладач та студент. Головний недолік таких систем - це обмеженість функціоналу зі сторони студента, як користувача. Адже важливими факторами покращення успішності студента є самоорганізація, систематизація навчальної діяльності та планування виконання завдань. Створений програмний застосунок допоможе студенту враховувати ці фактори.

Програма накопичує інформацію про навчальну діяльність студента за поточний період та зберігає за попередні. Наприклад: навчальні матеріали та додаткова інформація, посилання, роботи, оцінки за роботи та оцінки за результатами сесії, розклад навчання, інформацію про курс та дисципліни, план виконання робіт, порівняльні графіки по планах виконання. Таким чином система надає можливість створити ресурс про навчальну діяльність студента, його успішность та навчальні матеріали.

Окрім цього, програма слугуватиме помічником студента не тільки як інформаційний центр, але й як планувальник. Окрім можливостей планувальника, програма формує план виконання робіт враховуючи кінцевий термін та необхідний час для виконання роботи за рівномірностю часу виконання.

Накопичена інформація надає можливість створити інформаційну та файлову базу за весь час навчальної діяльності студента, що може бути використана у подальшій роботі чи навчанні.

1.1.1 Опис процесу діяльності

Об’єктом автоматизації є планування та систематизація навчальної діяльності студентів. Система призначена для підтримки та контролю навчальної діяльності студентів.

Схема структурна діяльності користувача наведена в графічному матеріалі. Система має одного актора.

Спочатку роботи в системі користувач повинен ввести інформацію про навчальний курс.

Вхідні дані на цьому етапі:

- навчальний заклад;
- факультет;
- номер курсу;
- номер семестру;
- дата початку курсу.

На вибір користувача система дозволяє заповнити навчальний розклад власноруч чи імпортувати його з сайту https://api.rozklad.org.ua за допомогою. представленого сайтом інтерфейсу. Після заповнення розкладу користувач може працювати з головним функціоналом. А саме:

- додавати роботи на день;
- додавати нагадування;
- створювати списки планових завдань по дисципліні;
- додавати детальну інформацію про дисципліну
- додавати файлові навчальні матеріали для завдань;
- додавати корисні посилання для завдань;
- додавати файлові матеріали виконаних завдань;
- додавати коментар та оцінку за завдання.

Додаючи роботи, що надходять часом, студент отримує план виконання даних робіт, який при бажані можна відредагувати. Система пропонує план

виконання запланованих робіт на наступні дні та зберігає список робіт на поточний день. В кінці тижня користувач може переглянути графік завантаженості по днях тижня, та порівняти з графіком попереднього тижня.

Після завершення певного навчального курсу користувач може переглядати накопичену інформацію по ньому, обравши зі списку пройдених курсів.

З пройдених курсів формується структурований по семестрах та дисциплінах архів, який за необхідністю можна вивантажити та переглянути.

1.1.2 Опис функціональної моделі

Єдиний користувач системи. Варіанти використання наведені в таблиці 1.1.

Таблиця 1.1 – Типи залежностей між варіантами використання

Актор	Варіант використання	Опис дії варіанта використання
Користувач	Введення інформації про курс навчання	Користувач має змогу вести інформацію про поточний курс навчання: факультет, номер курсу, номер семестру
	Заповнення розкладу	Користувач має змогу заповнити розклад занять власноручно
	Імпорт розкладу	Користувач має змогу імпортувати розклад занять з сайту https://api.rozklad.org.ua
	Додавання нагадувать	Користувач має смогу додавати нагадування у список нагадувань
	Додавання користних посилань	Користувач має можливість додавати корисні посилання до відповідного списку

Продовження таблиці 1.1

Актор	Варіант використання	Опис дії варіанта використання
	Додавання запланованих робіт	Користувач має можливсть додавати роботи, які потребують виконання. Для цього необхідно ввести назву та час виконання роботи, термін виконання роботи, та за бажанням опис роботи
	Заповнення інформації по дисципліні	Користувач має змогу ввести детальну інформацію по конкретній дисципліні (тип дисципліни, викладач, навчальні завдання, згідно освітньої програми)
	Додавання оцінки за навчальне завдання	Користувач має можливість ввести оцінку, яку він тримав за виконання завдання
	Додавання коментару за навчальне завдання	Користувач має можливість написати коментар до відповідного виконаного завдання
	Перегляд плану виконання робіт	Користувач має можливість переглядати створений програмою план виконання робіт. При використанні, програма зберігає поточний план на день
	Перегляд завантаженості роботами	Користувач має можливість переглянути графік завантаженості роботами на день протягом поточного тижня та переглянути відповідний графік з відповідним минулим тижнем

Продовження таблиці 1.1

Актор	Варіант використання	Опис дії варіанта використання
	Додавання навчальних матеріалів	Користувач має можливість додавати файлові навчальні матеріали по дисциплінам та навчальним завданням
	Додавання навчальних завдань	Користувач має можливість додавати файли та документи навчальних завдань
	Перегляд пройдених курсів	Користувач має можливість перегляду інформації по пройденим курсам, що була зібрана за період відповідного курсу
	Перегляд файлових даних	Користувач має можливість переглянути всі файли та документи, накопичені за час діяльності системи, згуртовані у вигляді ієрархічної системи папок
	Вивантаження даних	Користувач має можливість вивантажити всі файли та документи, накопичені за час діяльності системи, згуртовані у вигляді ієрархічної системи папок

У відповідність варіантам використання створена схема структурна варіантів використання, що наведена в графічному матеріалі.

1.2 Огляд наявних аналогів

«Schoolhouse» – програма-планувальник, розроблена під Mac OC X, створена для студентів. Призначена для планування процесу навчання та ведення домашніх завдань, звітів та різних проектів по навчальних процесах.

Застосування має наступний фунціонал:

- створення розкладу навчання;
- ведення запису домашніх завдань;
- сортування по заданим критеріям;
- створення подій по предметам;
- створення смарт-папок;
- створення переліку корисних посилань.

Застосування доступне для платного завантаження з сайту розробника та з Mac App Store.

«IStudiez» – мобільне застосування-планувальник для пристроїв сімейства Apple. Має подібний з вищерозглянутим застосунком функціонал, головна відмінність якого – наявність «хмарного» сервісу збереження даних. Тобто дані можна буде синхронізувати з різними девайсами користувача. Доступний для безкоштовного завантаження.

«Xerox DocuShare» – платформа керування документами та інформацією. Системи, створенні на базі цієї платформи мають такі функціональні можливості:

- управління збереженням даних та документів;
- управління web-контентом;
- управління доступом для захисту данних;
- командна робота;
- управління потоками робіт та процесами;
- інтеграція в корпоративне інформативне середовище;
- сканування та візуалізація;

– функція пошуку документів та даних.

На даній платформі базується побудова Єдиної Інформаційної Системи Навчального закладу, що потрібна для покращення якості та ефективності навчального процесу, а саме:

– покращити доступність та своєчасне отримання студентами навчально-методичних матеріалів і іншої інформації про хід навчального процесу;
– отримати засіб для стеження за ходом навчального процесу;
– створити зручне середовище для підтримки науково-дослідницької роботи;
– збільшити швидкість та ефективність виконання адміністративно-управничих задач;
– запобігти витоку інформації шляхом створення єдиної бази знань;
– оптимізувати вартість зберігання інформації та документів.

«Moodle» – навчальна платформа призначена для об’єднання педагогів, адміністраторів та студентів в одну надійну, безпечну та інтегровану систему для створення персоналізованого навчального середовища. У середовищі «Moodle» студенти отримують:

– доступ до навчальних матеріалів та засобів для спілкування і тестування «24 на 7»;
– засоби для групової роботи;
– можливість перегляду результатів дистанційного курсу студентом;
– можливість перегляду результатів проходження тесту;
– можливість спілкування з викладачем через особисті повідомлення, форум, чат;
– можливість завантаження файлів з виконаними завданнями;
– можливість використання нагадувань про події у курсі.

Відкрита система управління навчання, впроваджена в багатьох вищих навчальнпх закладах України.

Не потребує для своєї роботи жодного платного програмного забезпечення.

Ключові характеристики розглянутих аналогів приведені в таблиці 1.2.

Таблиця 1.2 – Порівняння аналогів з розробленим застосунком

Характеристика	Даний застосунок	IStudiez	Xerox DocuShare	Moodle	Schoolhouse
Безкоштовність	+	-	-	+	-
Зручність інтерфейсу	-	+	+	+	+
Кроссплатформеність	-	-	+	+	-
Захищеність інформації	-	+	+	+	+
Постійність інформації	+	+	-	-	+
Створення плану виконанняробіт	+	-	-	-	-

1.3 Постановка задачі

1.3.1 Призначення розробки

Призначенням системи є забезпечення контролю навчальної діяльності студентів та покращення процесу планування навчання студентів.

1.3.2 Цілі та задачі розробки

Цілями розробки є:

- накопити інформацію про навчальну діяльність студентів;
- покращити організацію виконання навчальних робіт студентів;
- спростити процес пошуку навчальної інформації та інформації з діяльності студентів.

Для досягнення поставлених цілей необхідно розв’язати наступні задачі:

- введення навчального розкладу;
- введення інформації за навчальними дисциплінами та додавання завдань;
- створення плану виконання поставлених робіт;
- створення графіку навантаженості днів за роботами;
- формування архіву поточних та попередніх завершених навчальних дисциплін.

Висновок до розділу

У розділі проаналізовано проедметну область, визначено актора системи. Приведено схеми структурні варіантів використання та діяльності.

Приведені аналоги та їх порівняння по критеріям. Описані цілі розробки та задачі, необіхдні для їх досягнення.

2 ІНФОРМАЦІЙНЕ ЗАБЕЗПЕЧЕННЯ

2.1 Вхідні та вихідні дані

Вхідні дані представлені у таблиці 2.1.

Таблиця 2.1 – вхідних даних.

Данні	Опис
Дані про курс навчання	Дані користувача, призначені для опису курсу (назва вищого навчального закладу,назва факультету,номер курсу, назва групи, рік початку курсу, рік закінчення курсу, номер семестру)
Дані про розклад навчання	Дані користувача або імпортовані для створення навчального розкладу(номер тижня навчання,день тижня, порядок пари, повна назва пари, П.І.Б. викладача, корпус та номер аудиторії)
Дані про дисципліну	Дані користувача по обраній дисципліні(назва дисципліни, тип дисципліни, коментар до дисципліни)
Дані по навчальним завданням	Дані користувача по завданням дисципліни (назва, коментар, оцінка)
Дані для плану робіт	Дані користувача для планування робіт (назва роботи, час виконання роботи, кінцевий термін виконання, опис роботи)
Дані по корисним посиланням	Назва корисного посилання, адрес корисного посилання
Дані по нагадуванням	Назва нагадування, опис нагадування

Вихідні дані представлені в таблиці 2.2.

Таблиця 2.2 – вихідні дані

Дані	Опис
Дані про виконаний план	Календарний порядок робіт, що виконувались протягом тижня
Дані статистики виконаного плану	Графік відображення щоденної навантаженості протягом відповідного тижня
Файлові дані	Накопичені протягом часу файлові дані, представлені у ієрархічному порядку

2.2 Опис структури бази даних

У базі даних Microsoft SQL Server[13] було сформовано 15 таблиць: «EducationalCourse», «Semester», «EducationalCourse_Semester», «Reminder», «Semester_Reminder», «PlanOfJobs», «Semester_PlanOfJobs», «Schedule», «Semester_Schedule», «Subject», «Semester_Subject», «Task», «Subject_Task», «Link», «Semester_Link». Опишемо призначення та структуру кожної із них:

EducationalCourse - таблиця, яка містить інформацію про курс навчання:

- IdEducationalCourse - ідентифікатор курсу навчання;
- InstituteName;
- FacultName – назва факультету;
- CourseNumber – номер курсу;
- NameOfGroup – назва групи;
- StartDate;
- EndDate.

Semester - таблиця, що містить номер семестру навчального курсу:

- IdSemester – ідентифікатор семестру;
- NumberOfSemester – номер навчального семестру.

EducationalCourse_Semester - таблиця, яка зв’язує семестри до відповідних курсів навчання. В одному курсі може бути декілька семестрів:

- IdCourse_Semester – ідентифікатор зв’язку курсу та семестру;
- IdSemester - ідентифікатор відповідного семестру;
- IdEducationalCourse - ідентифікатор відповідного курсу.

Reminder - таблиця, яка містить інформацію про нагадування користувача:

- IdReminder - ідентифікатор нагадування;
- NameOfReminder – назва нагадування;
- DataReminder – зміст нагадування.

Semester_Reminder – таблиця, що пов’язує нагадування з семестром. У семестрі може бути багато нагадувань:

- IdSemester_Reminder – ідентифікатор зв’язку курсу та семестру;
- IdSemester - ідентифікатор відповідного семестру;
- IdReminder - ідентифікатор відповідного нагадування.

PlanOfJobs - таблиця, яка містить інформацію про план робіт відного поточного дня:

- IdPlanOfJobs - ідентифікатор плану робіт;
- DataPlan – дані плану робіт відповідного дня;
- CurrentDayDate – дата дня, відносно якого створений план.

Semester_PlanOfJobs - таблиця, що пов’язує семестр та відповідний план робіт. У семестрі може існувати багато планів виконання робіт:

- IdSemester_PlanOfJobs - ідентифікатор зв’язку семестру та плану робіт;
- IdSemester - ідентифікатор відповідного семестру;
- IdPlanOfJobs - ідентифікатор відповідного плану робіт.

Schedule - таблиця, яка містить інформацію про розклад навчання:

- IdSchedule – ідентифікатор розкладу;
- NumberOfWeek – номер тижня розкладу
- DataSchedule – дані розкладу.

Semester_Schedule - таблиця, яка зв’язує семестр з відповідним навчальним розкладом. В одному семестрі може бути декілька розкладів:

- IdSemester_Schedule - ідентифікатор зв’язку семестру та навчального розкладу;
- IdSemester - ідентифікатор відповідного семестру;
- IdSchedule - ідентифікатор відповідного навчального розкладу.

Subject - таблиця, яка містить інформацію про навчальну дисципліну:

- IdSubject – ідентифікатор дисципліни;
- NameOfSubject – назва дисципліни;
- SubjectData – дані дисципліни.

Semester_Subject - таблиця, яка зв’язує навчальну дисципліну з відповідним семестром. В одному семестрі може бути декілька дисциплін:

- IdSemester_Subject - ідентифікатор зв’язку семестру та навчальної дисципліни;
- IdSemester - ідентифікатор відповідного семестру;
- IdSubject - ідентифікатор відповідної навчальної дисципліни.

Task - таблиця, яка містить інформацію про навчальні завдання:

- IdTask – ідентифікатор навчального завдання;
- NameTask – назва навчального завдання;
- TaskData – дані завдання.

Subject_Task - таблиця, яка зв’язує навчальну дисципліну та навчальні завдання з цієї дисципліни. В одній дисципліні може бути багато завдань:

- IdSubject_Task - ідентифікатор зв’язку навчальної дисципліни та завдання з цієї дисципліни;
- IdTask - ідентифікатор відповідного навчального завдання;

- IdSubject - ідентифікатор відповідної навчальної дисципліни.

Link - таблиця, яка містить корисні посилання користувача:

- IdLink – ідентифікатор посилання;
- NameOfLink – назва посилання;
- DataOfLink – адрес посилання.

Semester_Link - таблиця, яка зв’язує семестр з корисними посиланнями користувача. В одному семестрі може бути багато корисних посилань:

- IdSemester_Link - ідентифікатор зв’язку семестру та корисного посилання;
- IdSemester - ідентифікатор відповідного семестру;
- IdLink - ідентифікатор відповідного корисного посилання.

Схема структурна бази даних наведена у графічному матеріалі.

Висновок до розділу

В розділі описані вхідні та вихідні дані, наведені у вигляді таблиць. Описані таблиці бази даних та їх поля. Наведена схема бази даних, що показує зв’язки між таблицями.

3 МАТЕМАТИЧНЕ ЗАБЕЗПЕЧЕННЯ

3.1 Змістовна постановка задачі

– Досліження проблем впорядкування або визначення виконання сукупності робіт, створення плану виконання робіт є частиною дисципліни Дослідження операцій, а саме Теорії розкладів[15].

Щоб класифікувати задачі теорії розкладів, використовуватимо запис *A/B/C/D,* у якому*:*

– *A* - характеристика надходження робіт як процесу. У динамічних задачах представлена функцією розподілу часу між надходженнями робіт. У статичних задачах їй відповідає число робіт, що одночасно поступили, за умови, що для цього нічого не було обумовлено спеціально, наприклад, якщо в записі у відповідному місці стоїть n – це означає довільне кінцеве число робіт.

– *B* - характеристика числа машин(пристроїв) в системі. Наприклад, якщо в записі у відповідному місці стоїть m - це означає довільне число машин(пристроїв).

– *C* - характеристика порядку виконання робіт машинами(пристроями). Наприклад, якщо в записі у відповідному місці стоїть R, то це відповідає випадковій системі; якщо F - конвеєрній, та довільній. Якщо *G*. Якщо машина в системі одна, третій параметр не має змісту, а отже не розглядається.

– *D* - характеристика критерію розкладу.

Узагальнено задача теорії розкладів описується наступним чином: $n/m/G/F_{\max}$ та має наступний зміст: впорядкувати *n* робіт в довільній системі з машин так, щоб мінімізувати максимальну тривалість проходження роботи[15].

Головна проблема конкретної задачі, що розглядається в дипломному проекті – це те, що кінцеве число робіт невідоме, якщо розглядати статичний випадок задач, та неможливо задати функцію розподілу часу між надходженням робіт, якщо розлядати задачу як динамічний випадок.

Саме часова характеристика задач теорії розкладів об’єднує їх в клас, що суттєво різниться від об'ємних економічних задач. В останніх задачах потрібно дати відповідь на питання «що і скільки виробляти?». В задачах теорії розкладів необхідно визначити в якій послідовності та коли виконувати роботи. Наведені вище відмінності в суті завдань визначають відмінності у можливостях та методах їх розв’язання.

В порівнянні з задачами об’ємного характеру, які мають досить потужний математичний апарат, задачі теорії розкладів мають в значно меншій мірі розвинений апарат розв’язання.

Поганий розвиток точних методів розв’язання задач теорії розкладів сприяв розвитку наближених методів, які у свою чергу отримують прийнятні розв’язки за порівняно невеликі витрати коштів та часу. Наближені методи діляться на імовірнісні та евристичні.

Евристичні алгоритми засновані на прийомі зниження вимог, що полягає у відмові пошуку оптимального розв’язку за прийнятний час. Дані алгоритми керуються різними розумними міркуваннями, не маючи строгих обґрунтувань.

Часто використовується метод локального пошуку, в якому наперед обрана множина перестановок використовується для послідовного покращення початкового розв’язку, допоки існує таке покращення, інакше досягнуто локального оптимуму.

Також серед евристичних методів розв’язку задач теорії розкладів існує метод , що полягає в формуванні правил переваг. Тобто для кожної роботи з множини робіт формуються так звані переваги та пріоритети, за якими обиратиметься краща робота[15].

Задача, що розглядається в дипломному проекті розв’язується алгоритмом, що являє собою перестановки елементів підмножин у різні підмножини з додаванням правил переваг та пріоритетів, крім того додаванням обмежень на різних кроках алгоритму.

Задача має такі поняття як пріоритети та множина робіт. Множина робіт невпорядкована. Кожна робота характеризується часом виконання та дедлайном. Виконується відношення послідовності виконання робіт (користувач не може виконувати паралельно декілька робіт). Згідно логіки планування виконання робіт, пов'язаних з навчальною діяльністю, система перераховує план при кожному наступному дні. Надходження робіт визначається користувачем.

Мета задачі полягає в складані плану виконання робіт, у якому сума годин виконання робіт на день була наближеною до всіх відповідних значень (тобто перерозподіл робіт, з метою урівнювання завантаженості на день протягом всього плану).

3.2 Математична постановка задачі

Дано множину робіт $A = \{a_1, a_2, \dots, a_m\}$. Роботи виконуються у послідовному порядку. Роботи мають відомі часовий обсяг виконання t_{a_i}та дедлайн d_{a_i} (заданий термін виконання роботи). Відповідно до дедлайну, роботи відрізняються пріоритетом w_{a_i}. Крім характеристик робіт існує поняття поточного дня. Чим ближче дедлайни робіт до поточного дня, тим вищий пріоритет роботи, наприклад, якщо дедлайн деякої роботи - наступний день, то її пріоритет рівний $w_{a_i} = 1$(найвищий пріоритет).

Відповідно, якщо роботи потрібно виконати через 2 дні, їх пріоритет 2 і т.д.

Введемо зміну x_{ij}, яка приймає наступні значення:

$$x_{ij} = \begin{cases} 1, \text{якщо } i - \text{й елемент належить } j \text{ підмножині;} \\ 0, \text{якщо } i - \text{й елемент не належить } j \text{ підмножині,} \end{cases}$$

де i = 0, … m;

j = 0, … n; n – кількість днів, що мають дедлайни робіт.

Обмеження:

$$\sum_{j=0}^{n} x_{ij} = 1$$

Введемо додаткову характеристику підмножин вагу підмножини:

$$\gamma_j = \sum_{i=0}^{m} t_{a_i} x_{ij}$$

Тоді цільова функція буде мати вигляд:

$$\sum_{i=0}^{n-1} \sum_{k=i+1}^{n} |\gamma_i - \gamma_k| \to min$$

Підставимо формулу ваги підмножини. Отримали математичну модель задачі:

$$\sum_{i=0}^{n-1} \sum_{k=i+1}^{n} \left| \sum_{l=0}^{m} t_{a_l} x_{li} - \sum_{l=0}^{m} t_{a_l} x_{lk} \right| \to min$$

Обмеження:

$$\sum_{j=0}^{n} x_{ij} = 1$$

де $x_{ij} \in \{0; 1\}$;

$i = 0, \dots m$;

$j = 0, \dots n$.

Позбавимось від модуля в цільовій функції. Для цього замінимо вираз

$$\left| \sum_{l=0}^{m} t_{a_l} x_{li} - \sum_{l=0}^{m} t_{a_l} x_{lk} \right|$$

на u_{ik}.

Отримаємо:

$$\sum_{i=0}^{n-1}\sum_{k=i+1}^{n}|u_{ik}| \to min$$

Розкривши модуль, отримаємо групи обмежень, які накладаються на u_{ik}:

$$u_{ik} \geq \sum_{l=0}^{m} t_{a_l} x_{li} - \sum_{l=0}^{m} t_{a_l} x_{lk} ,$$

$$u_{ik} \geq -\left(\sum_{l=0}^{m} t_{a_l} x_{li} - \sum_{l=0}^{m} t_{a_l} x_{lk}\right).$$

Зведемо другу групу обмежень до вигляду "≤", отримаємо:

$$u_{ik} \geq \sum_{l=0}^{m} t_{a_l} x_{li} - \sum_{l=0}^{m} t_{a_l} x_{lk} ,$$

$$-u_{ik} \leq \sum_{l=0}^{m} t_{a_l} x_{li} - \sum_{l=0}^{m} t_{a_l} x_{lk} .$$

Перенесемо змінні в ліву частину, а вільні в праву:

$$u_{ik} - \sum_{l=0}^{m} t_{a_l} x_{li} + \sum_{l=0}^{m} t_{a_l} x_{lk} \geq 0 ,$$

$$-u_{ik} - \sum_{l=0}^{m} t_{a_l} x_{li} + \sum_{l=0}^{m} t_{a_l} x_{lk} \leq 0 .$$

Отже, математична модель задачі така:

$$\sum_{i=0}^{n-1}\sum_{k=i+1}^{n} u_{ik} \to min$$

$$u_{ik} - \sum_{l=0}^{m} t_{a_l} x_{li} + \sum_{l=0}^{m} t_{a_l} x_{lk} \geq 0 ,$$

$$-u_{ik} - \sum_{l=0}^{m} t_{a_l} x_{li} + \sum_{l=0}^{m} t_{a_l} x_{lk} \leq 0 ,$$

$$\sum_{j=0}^{n} x_{ij} = 1 ,$$

де $x_{ij} \in \{0; 1\}$,

$i = 0, \dots m$,

$j = 0, \dots n$.

Отримали задачу лінійного програмування.

Проте, при цьому дана модель задачі має наступні правила переваг та додаткові обмеження:

Робота j переважає роботу k якщо:

- її тривалість менша:

$$a_j \leq a_k ;$$

- її пріоритет менше число(найвищий пріоритет – 0):

$$w_j \leq w_k .$$

Роботи з пріоритетом 1 не можуть бути переміщенні до іншої підмножини.

При переході до нового поточного дня, підмножинна попереднього видаляється, перераховуються пріоритети старих робіт, відносно нового поточного дня та можливо надходять нові роботи.

3.3 Опис алгоритму розв'язку задачі

КРОК 1. Розставляємо пріоритети підмножин (днів) у порядку наближення до поточного дня (0-пріоритет).

КРОК 2. Переміщаємо всі роботи кожної підмножини в підмножину з пріоритетом, меншим на одиницю (в підмножину, що знаходиться «лівіше»). Підмножини з пріоритетом 0 та ≥ 6 ігноруються.

КРОК 3. Обчислюємо суму годин на виконання робіт у кожній підмножині. Сортуємо роботи у порядку зростання тривалості виконання.

КРОК 4. Для найбільшої підмножини починаємо проходити по всіх елементах.

КРОК 4.1. Додаємо обраний елемент у лівішу підмножину.

КРОК 4.2. Якщо відбується покращення цільової функції при переміщенні, то залишаємо елемент у лівішій підмножині. Інакше обираємо наступний елемент підмножини. Повторюємо процес для всіх елементів підмножини.

Крок 4.3. Перераховуємо суми годин на виконання робіт у відповідних підмножинах.

КРОК 5 Знову обираємо найбільшу підмножину та проходимо по всіх елементах. Повторюємо КРОК 4 – КРОК 4.3.

3.4 Приклад використання алгоритму розв'язку задачі

Маємо наступні умови, зображені на рисунку 3.1:

	Пн	Вт	Ср	Чт	Пт	Сб	Нд
Пріоритет	0	1	2	3	4	5	6
Роботи		1	2	2	1	0,5	2
(години)		2	1		2		3
			2		0,5		
					1,5		

Рисунок 3.1 - Початкові умови

Тобто, наприклад, користувач в понеділок додав до календаря наступні роботи з часом виконання з вказаним днем дедлайну. Пріоритет поточного дня – 0, всі наступні дні мають пріоритет, збільшений на 1.

Першим чином зсунемо вліво всі роботи. Ігноруємо роботи з пріоритетом більше 6. Відсортуємо роботи у порядку зростання в кожному дні (елемент підмножини з меншою тривалістю має більший пріоритет). Порахуємо суму годин виконання робіт на день та оберемо день, з найвищою завантаженістю. Такими є вівторок та четвер. Обираємо четвер, так як його пріоритет менший (число пріоритету більше), дивитися рисунок 3.2.

	Пн	Вт	Ср	Чт	Пт	Сб	Нд
Пріоритет	0	1	2	3	4	5	
	1/1	1/2	2/3	0,5/4	0,5/5		
	2/1	2/2		1/4			
		2/2		1,5/4			
				2/4			
сума годин	3	5	2	5	0.5		

Рисунок 3.2 - Хід виконання алгоритму

Модуль різниці завантаженостей (ваг підмножин) середи та четверга рівна 3. Починаємо розглядати елементи обраної підмножини (четвер). Якщо перемістити роботу 0.5/4, відбудеться покращення цільової функції (модуль різниці завантаженностей підмножин стане рівним 2). Переміщаємо даний елемент до лівішої підмножини, дивитись рисунок 3.3.

	Пн	Вт	Ср	Чт	Пт	Сб	Нд
Пріоритет	0	1	2	3	4	5	
	1/1	1/2	2/3		0,5/5		
	2/1	2/2	0,5/4	1/4			
		2/2		1,5/4			
				2/4			
сума годин	3	5	2,5	4,5	0.5		

Рисунок 3.3 - Хід виконання алгоритму

Обираємо роботу 1/4, при перенесенні її до лівішої підмножини відбулося покращення цільової функції (модуль різници завантаженості став рівний 0), переносимо її, отримуємо наступний план, дивитись рисунок 3.4.

	Пн	Вт	Ср	Чт	Пт	Сб	Нд
Пріоритет	0	1	2	3	4	5	
	1/1	1/2	2/3		0,5/5		
	2/1	2/2	0,5/4				
		2/2	1/4	1,5/4			
				2/4			
сума годи	3	5	3,5	3,5	0.5		

Рисунок 3.4 - Хід виконання алгоритму

Обираємно наступну найбільшу підмножину (вівторок), дивитись рисунок 3.5.

	Пн	Вт	Ср	Чт	Пт	Сб	Нд
Пріоритет	0	1	2	3	4	5	
	1/1	1/2	2/3		0,5/5		
	2/1	2/2	0,5/4				
		2/2	1/4	1,5/4			
				2/4			
сума годин	3	5	3,5	3,5	0.5		

Рисунок 3.5 - Хід виконання алгоритму

Обираємо елемент 1/2, переносимо його до лівішої підмножини, отримуємо покращення цільової функції (модуль різниці завантаженностей рівний нулю), переносимо даний елемент, дивитись рисунок 3.6.

	Пн	Вт	Ср	Чт	Пт	Сб	Нд
Пріоритет	0	1	2	3	4	5	
	1/1		2/3		0,5/5		
	2/1	2/2	0,5/4				
	1/2	2/2	1/4	1,5/4			
				2/4			
сума годин	4	4	3,5	3,5	0.5		

Рисунок 3.6 - Хід виконання алгоритму

Обираємо множини, які ще не порівнювали з лівішими. Така одна – п’ятниця, тому що множини з пріоритетом 0 ігноруються. Так як при перестановці елемента у лівішу множину не відбудеться покращення цільової функції, залишаємо елемент на місці. Додаємо до плану роботи з пріоритетом 6 та більше. Маємо остаточний план виконання робіт, дивитися рисунок 3.8.

	Пн	Вт	Ср	Чт	Пт	Сб	Нд
Пріоритет	0	1	2	3	4	5	6
	1/1		2/3		0,5/5		2/6
	2/1	2/2	0,5/4				3/6
	1/2	2/2	1/4	1,5/4			
				2/4			
сума годин	4	4	3,5	3,5	0.5		

Рисунок 3.7 - Отриманий план виконання робіт

Для наступного дня надходять нові роботи, перераховуються пріоритет відповідно нового поточного дня, початкова множина робіт на новий поточний день порожня.

Висновок до розділу

В даному розділі дипломного проекту було розглянуто змістовну та математичну постановку задачі. Також був описаний алгоритм розв’язку задачі та наведений приклад використання алгоритму розв’язку з поясненням.

4 ПРОГРАМНЕ ТА ТЕХНІЧНЕ ЗАБЕЗПЕЧЕННЯ

4.1 Засоби розробки

Програма розроблялась в програмному середовищі Microsoft Visual Studio 2012, за допомогою мови програмування C#. Візуальна складова та логіка застосунку розроблена за допомогою WPF (Windows Presentation Foundation) – системи для побудови клієнтських застосунків Windows з візуальними можливостями взаємодії з користувачем[8].

В основі WPF взято векторну систему візуалізації, що не залежить від роздільної здатності пристрою виведення та створена з врахуванням можливостей сучасного графічного обладнання. WPF надає засоби для створення візуального інтерфейсу, включаючи мову XAML (eXtensible Application Markup Language), елементи керування, прив’язку даних, макети, двовимірну та тривимірну графіку, анімацію, стилі, шаблони, документи, текст, мультимедіа та оформлення. Графічна технологія , що лягла в основу WPF, являється DirectX, яка має кращу швидкодію серед аналогів, отриману за рахунок апаратного прискорення[8].

XAML являє собою мову декларативного опису інтерфейсу, основаного на XML (eXtensibleMarkup Language), в якій реалізовано модель розділу коду та дизайну, що дозволяє програмісту та дизайнеру кооперуватися. Крім того, в ній є вбудована підтримка стилів елементів, а самі елементи легко розділити на елементи керування другого рівня, які, в свою чергу, розділяються до рівня векторних фігур і властивостей/дій. Це надає змогу легко задавати стиль для будь-якого елемента[11].

Засобом роботи з базою даних являється ADO.NET Entity Framework, що являє собою об’єктно-орієнтовану технологію доступу до даних та технологію об’єктно-зв’язкового мапінгу для платформи .NET Framework [10].

Entity Framework являється продовженням технології Microsoft ActiveX Data та надає можливість роботи з базами даних через об'єктно-орієнтований код C#. Цей підхід надає ряд істотних переваг: вам не потрібно думати про код доступу до даних, не потрібно знати деталей роботи з СКБД SQL Server та синтаксису мови запитів T-SQL. Замість цього надається можливість роботи з таблицями баз даних як з класами C#. З полями цих таблиць як з властивостями класів. А синтаксис SQL – запитів замінений на зручніший підхід з LINQ[10].

При роботі з Entity Framework вам надаються великі можливості по створенню моделі бази даних за допомогою Visual Studio. Вам надаються три підходи для проектування бази даних:

- Database-First – коли спочатку створюється база даних за допомогою різноманітних інструментів. А потім генерується EDMX – модель бази даних;
- Model-First – коли спочку створюється EDMX – модель бази даних, а потім на її основі генерується база даних;
- Code-First - модель EDMX не використовується взагалі та вручну налаштовуються класи C# об'єктної моделі. Даний підхід підтримує як генерацію існуючих класів із існуючої бази даних, так і створення бази даних із створеної власноруч моделі об'єктів C#[10].

4.2 Вимоги до технічного забезпечення

4.2.1 Загальні вимоги

Так як даний продукт являє собою desktop-застосунок на платформі WPF, користувачі повинні використовувати комп'ютер на базі операційної системи Windows(7 та вище) та з наступними конфігураціями:

– процесор – Intel Pentium 2 ГГц і вище;
– мінімальний об'єм оперативної пам'яті – 512 Мб;

– об’єм вільної пам’яті на пристрої від 50 мб;
– для деяких функцій наявність інтернет зв'язку.

4.3 Архітектура програмного забезпечення

4.3.1 Опис архітектури програмного забезпечення

Архітектура застосунку реалізована на архітектурному патерні MVVM (Model-View-ViewModel), що відділяє логіку застосунку від візуального уявлення. Складається з трьох компонентів: моделі (Model), моделі представлення (ViewModel) та представлення (View)[9].

Model – описує дані, що використовуються в застосунку. Модель містить логіку, що напряму пов’язана з цими даними. Не містить ніякої логіки, пов’язаної з відображенням даних і взаємодії з візуальними елементами управління[9].

View – визначає візуальний інтерфейс, через який користувач взаємодіє з додатком. В WPF це код в XAML, що визначає інтерфейс у вигляді кнопок, текстових полів і інших візуальних елементів[9].

ViewModel – зв’язує модель та представлення через механізм прив’язки даних. Якщо в моделі зміняться значення властивостей, при реалізації моделлю інтерфейса INotifyPropertyChanged автоматично йде зміна відображуваних даних в представленні, хоча напряму модель та представлення не зв’язані. Також містить логіку по отриманню даних з моделі, які потім передаються в представлення, визначає логіку по оновленню даних в моделі. Так як елементи представлення не використовують події (Event), то представлення взаємодіє з ViewModel за допомогою команд[9].

Отже, за допомогою патерну MVVM відбувається функціональне розділення застосунку на три компоненти, які простіше розробляти та тестувати.

Загальна діаграма патерну MVVM представлена на рисунку 4.1.

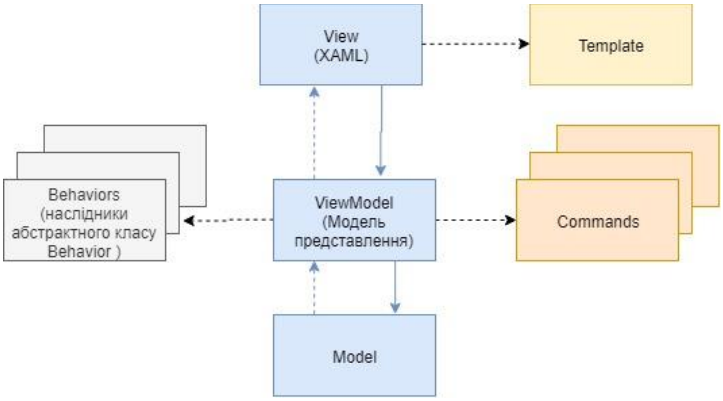


Рисунок 4.1 – Загальна діаграма патерну MVVM

Структурна схема класів наведена в графічному матеріалі. Опис компонентів діаграми класів наведений в таблиці 4.1.

Таблиця 4.1 – Опис компонентів діаграми класів

Компонент	Опис компоненту
View	
CurrentCourse	Вікно поточного курсу
PlanOfJobs	Клас відповідної таблиці бази даних
ViewModel	
SetDivisionTaskInputsViewModel	Логіка представлення зв’язана з вхідними даними
SetDivisionTaskOutputsViewModel	Логіка представлення зв’язана з вихідними даними
SetDivisionTaskSolverViewModelMediator	Логіка представлення, яка реалізує взаємодію між вхідними і вихідними даними
RelayCommand	Шаблонний клас команди для прив’язки даних
NotificationBase	Базовий клас для ViewModel
Model	
SetDivisionTaskInputs	Клас, який інкапсулює вхідні дані
SetDivisionTaskOutputs	Клас, який інкапсулює вихідні дані
SetDivisionTaskSolver	Клас, який має методи вирішення задачі

Продовження таблиці 4.1

Компонент	Опис компоненту
SetDivider-TaskSolution	Клас, який інкапсулює дані про розв’язки
RearegmentCom-mand	Команда, яка інкапсулю перестановку (базовий клас)
DonateReareg-ment	Команда, яка інкапсулю перестановку передавання
ExchangeCom-mand	Команда, яка інкапсулю перестановку обміну

У таблиці 4.2 наведено специфікація методів.

Таблиця 4.2 – Опис методів класів

Назва класу	Назва методу	Дія
PlanOfJobs	OnConfiguring()	Метод, що дозволяє створювати параметри підключення до бази даних. Для їх створення він викликає метод UseSqlServer в який передається рядок підключення
NotificationBase	RaisePropertyChanged()	Коли об’єкт класу змінює значення своєї властивості, метод сповіщає систему про зміну властивості, через подію PropertyChanged

Продовження таблиці 4.2.

Назва класу	Назва методу	Дія
NotificationBase	SetProperty(ref_delt, value)	Метод, що присвоює значення з другого переданого параметра в значення першого переданого параметра а також викликає метод RaisePropertyChanged()
RelayCommand	CanExecute(object parameter)	Метод, що ставить прапорець. Якщо він true, команда ввімкнута та буде виконуватись, якщо false, команда буде відключена
	Execute(object parameter)	Метод, призначений для зберігання логіки команди. Містить метод CanExecute()
SetDivisionTaskSolver	DoExchanges()	Метод, що переміщає елемент в іншу підмножину
	LazySetUp()	Метод, що сортує та переміщує початкові підмножини елементів
	NumberOfBiggestSolution()	Метод, що знаходить кількість найбільних підмножин

Продовження таблиці 4.2

Назва класу	Назва методу	Дія
SetDivisionTaskSolver	Solve()	Метод, що виконує алгоритм
	TryDonate()	Метод, що перевіряє стан цільової функції, при переміщенні елемента у іншу підмножину
	Clone()	Метод, що створює копію об’єкта
	CountWeight()	Метод, що рахує ваги підмножин після кожної ітеорації

Структурна схема послідовності наведена в графічному матеріалі.

Створення плану виконання робіт виконується наступним чином: користувач вводить роботи, вказуючи час виконання та кінцевий термін. Користувач взаємодіє з програмою через графічний інтерфейс, дані робіт вводяться у вікні CurrentCourse. View взаємодіє з ViewModel об’єктами за допомогою команд. Об'єкти ViewModel зв’язують модель та представлення через механізм прив’язки даних (Data binding) за допомогою інтерфейса INotifyPropertyChanged, також містять логіку отримання даних з моделі, які потім передаються в представлення та визначають логіку оновлення даних в моделі. Об’єкти Model описують дані, що використовуються в застосунку та логіку їх взаємодії.

Схема структурна компонентів наведена в графічному матеріалі. Програма має декілька структурних компонентів, зображених на схемі структурній компонентів, також приведено їх залежності один з одним.

Висновок до розділу

В даному розділі було розглянуто засоби та мови програмування, за допомогою яких було реалізовано програмний продукт. Приведені структурні схеми класів, послідовності та розгортання. В таблиці приведено опис компонентів діаграми класів.

5 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

5.1 Керівництво користувача

Щоб показати процес взаємодії додатку з користувачем наведемо приклади виконання додатку. На рисунках 5.1-5.24 приведені скріншоти роботи з програмою.

Після запуску застосунку з'являється початкове вікно на якому розташовані 3 кнопки, при натисненні на які відбувається наступний сценарій виконання програми (дивитись рисунок 5.1.):

- кнопка «To current course». При її натискані відображається вікно поточного курсу;
- кнопка «History». При її натискані відображається вікно перегляду історії курсів;
- кнопка «Documents». При її натисканні відображається вікно перегляду накопичених файлів.

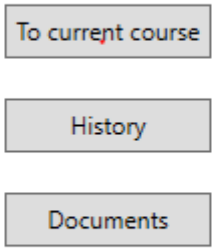


Рисунок 5.1 - Початкове вікно

При першому запуску додатку, при натискані кнопок «History» та «Documents» з’явиться діалогове вікно з повідомленням, про відсутність даних, дивитись рисунок 5.2.

Рисунок 5.2 - Діалогове вікно

А при натискані кнопки «To current course» з’явиться діалогове вікно з повідомленням про відсутність курсу, дивитись рисунок 5.3.

Рисунок 5.3 - Діалогове вікно

При натискані клавіші «OK» з’явиться вікно додавання курсу, дивитись рисунок 5.4.

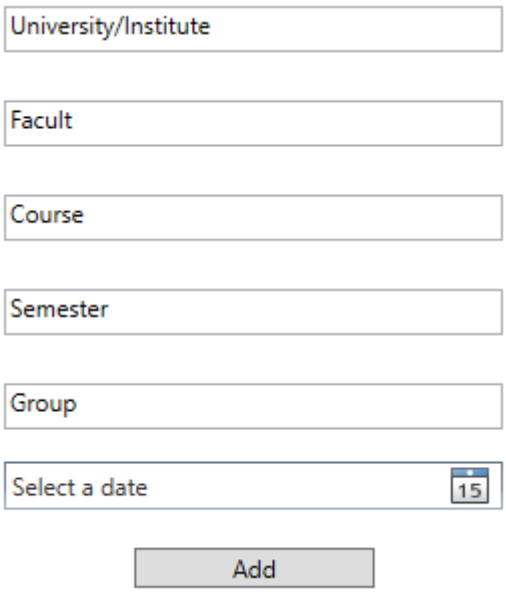


Рисунок 5.4. - Вікно додавання курсу

Після заповнення відповідних текстових полів та натиснувши кнопку «Add» з'явится вікно створеного поточного курсу.

З цього моменту вікно поточного курсу буде появлятись весь час, після натискання кнопки «To current course» початкового вікна додатку. Вікно поточного курсу виглядає так, дивитись рисунок 5.5.

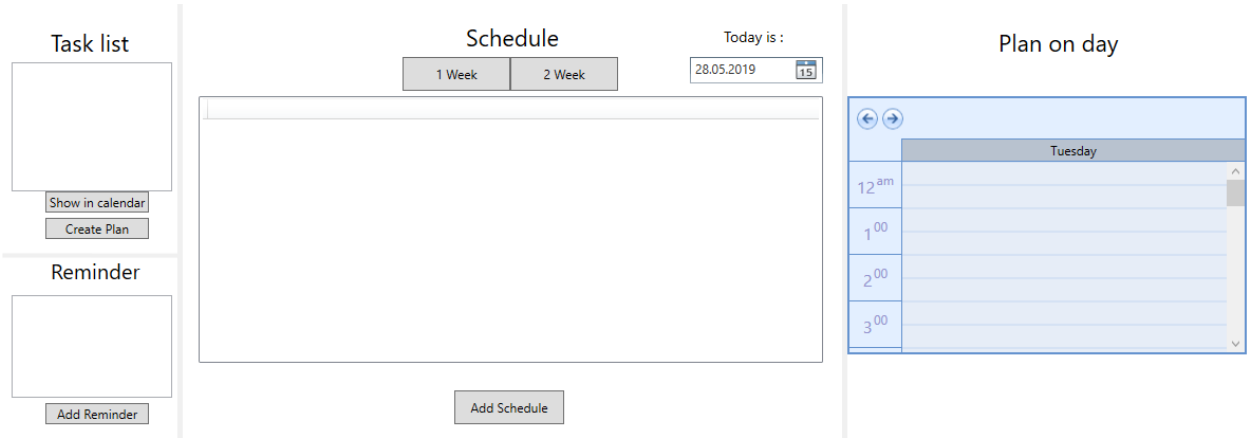


Рисунок 5.5 – Вікно поточного курсу

У вікні поточного курсу відображається різного типу інформація. В списку робіт (Task list) відображається список запланованих вами робіт, щоб додати роботу до списку, необхідно натиснути кнопку «Show at calendar», відкриється вікно календаря, дивитись рисунок 5.6.

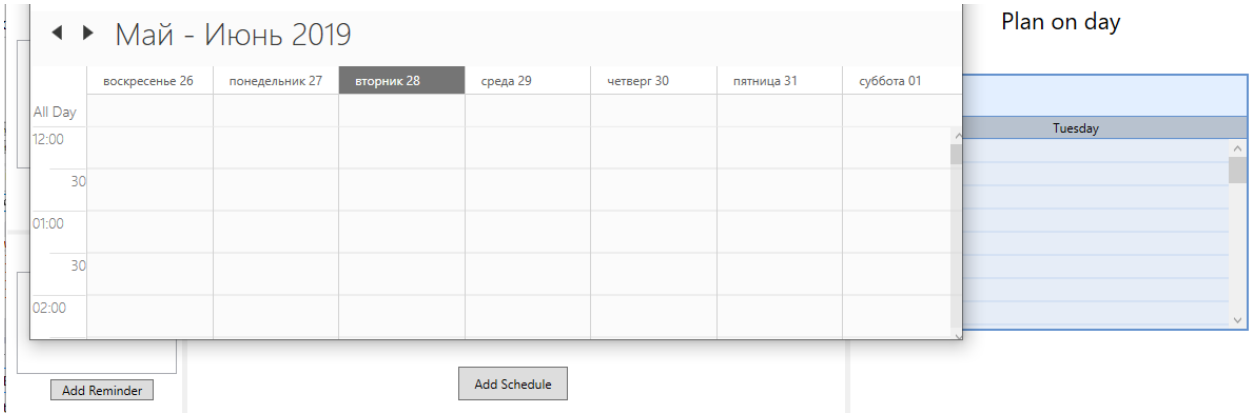


Рисунок 5.6 – Вікно календаря робіт

Щоб додати роботу, натисність праву клавішу мишки в області потрібного вам дня тижня, та натисніть на «AddNew» , дивитись рисунок 5.7.

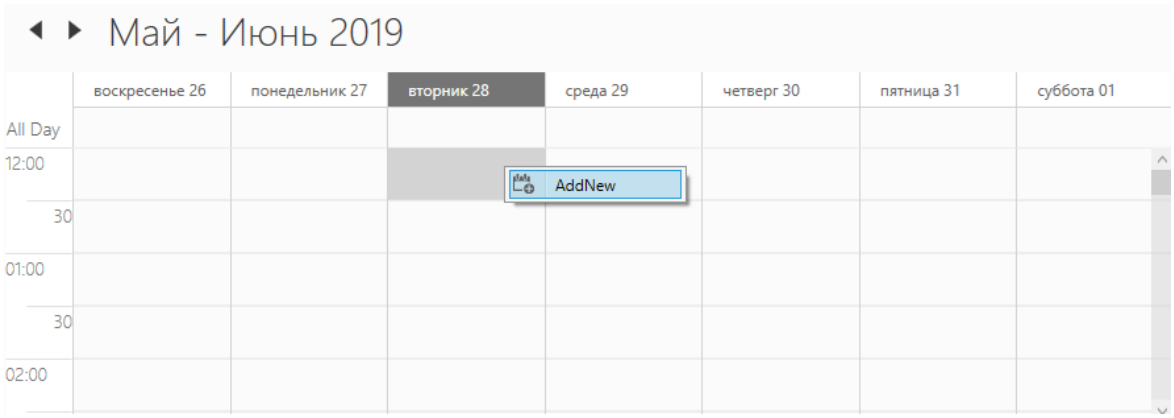


Рисунок 5.7 – Вікно календаря робіт. Додавання роботи

При натисканні, з’явиться вікно додавання роботи, дивитись рисунок 5.8.

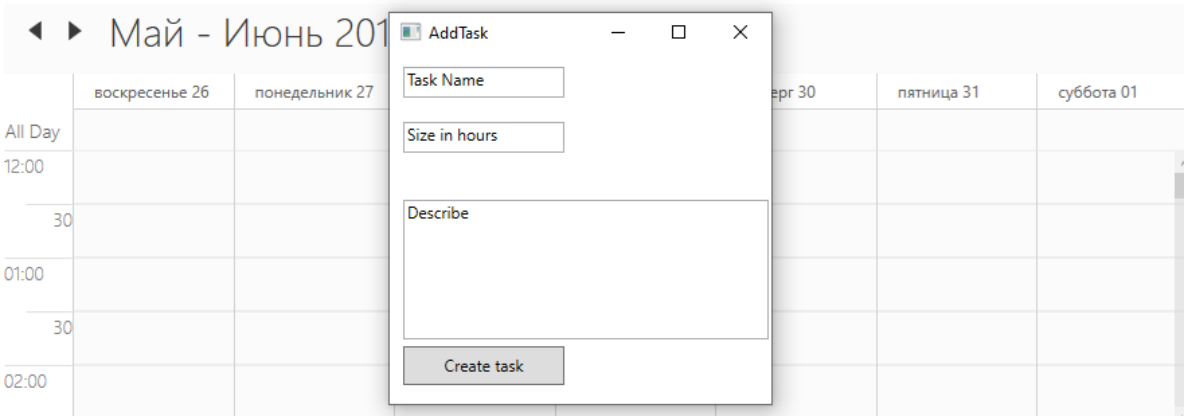


Рисунок 5.8 – Вікно додавання роботи

Ввівши необхідну інформацію та натиснувши клавішу «Create task» ви

створите нову роботу. Вона відображатиметься як на календарі так і у списку робіт(Task list). Такі значення як початок та кінець роботи не вказуються. Роботи відображаються у порядку появи, дивитись рисунок 5.9.

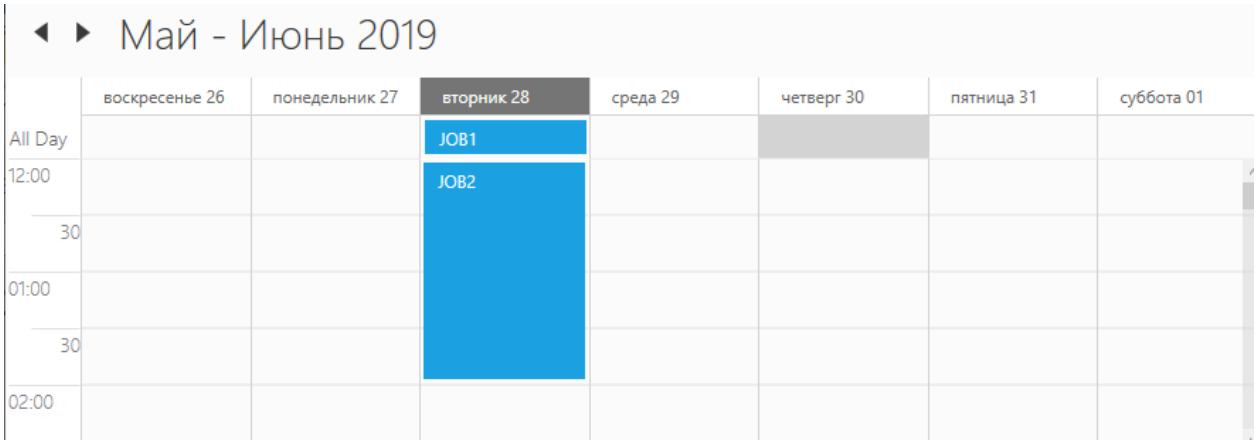


Рисунок 5.9 – Вікно календаря робіт з доданими роботами

У списці робіт (Task list) з'являються також, з описом назви роботи та її дедлайну, дивитись рисунок 5.10.

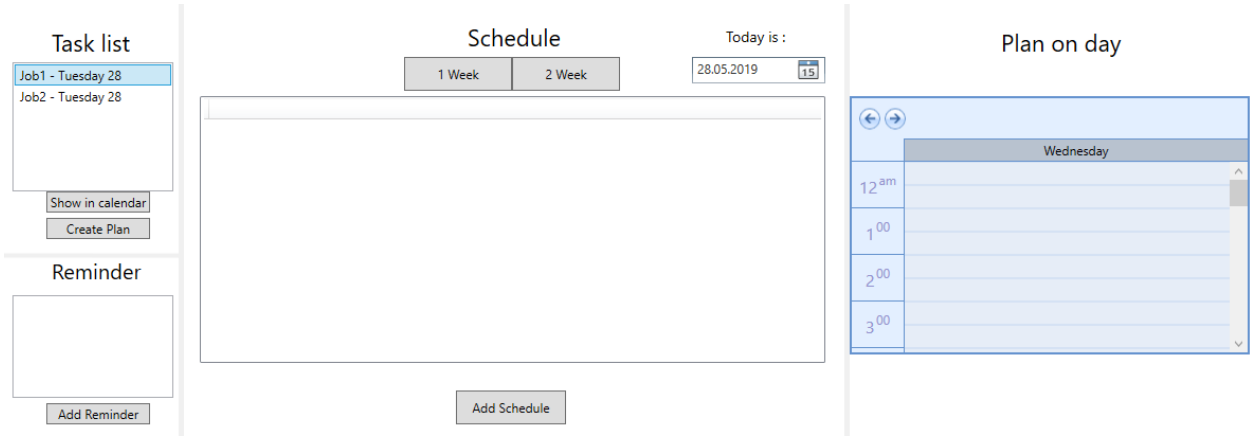


Рисунок 5.10 – Вікно календаря робіт з доданими роботами

Натиснувши на відповідну роботу, можна переглянути детальну інформацію про обрану роботу (назву, дедлайн та його день, час виконання, опис), дивитись рисунок 5.11.

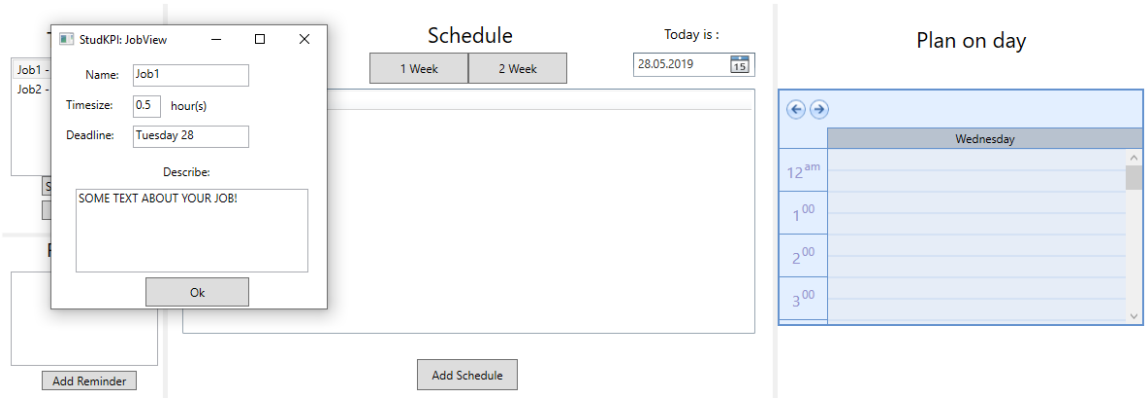


Рисунок 5.11 – Вікно з описом конкретної роботи

Записавши достатню кількість робіт для виконання, натиснувши клавішу «Create Plan» вам запропонується план виконання робіт, що відображатиметься в правій частині інтерфейсу поточного курсу, під назвою «Plan on day» , дивитись рисунок 5.12.

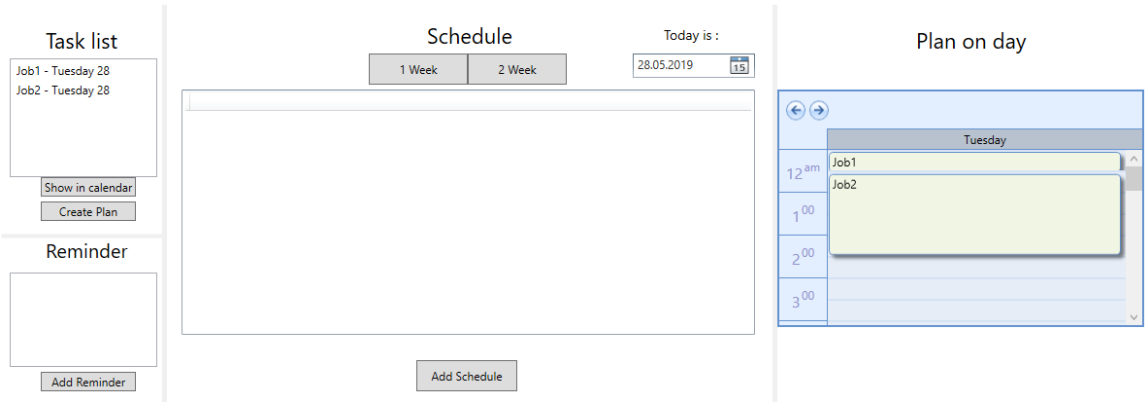


Рисунок 5.12 – Вікно з планом порядку виконання робіт

Роботи не мають прив’язки до конкретних годин, тому їх можна рухати, але при цьому зберігається їх послідовність. Натиснувши на стрілочку можна переглядати інші дні тижня та відповідно інші плани на обраний день, якщо такі є.

Натиснувши на клавішу «Add Reminder» ви створюєте нагадування, які відображаються весь час у відповідному списку «Reminder» , дивитись рисунок 5.13.

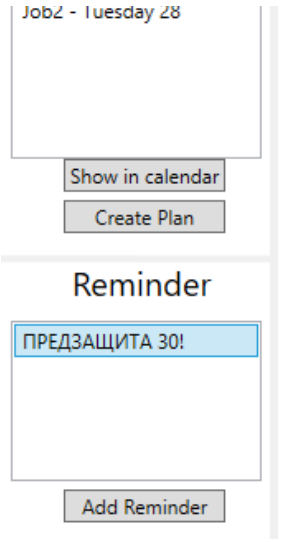


Рисунок 5.13 – Вікно нагадуванням

Переглянути весь текст, та видалити нагадування можна, натиснувши на відповідне нагадування, дивитись рисунок 5.14.

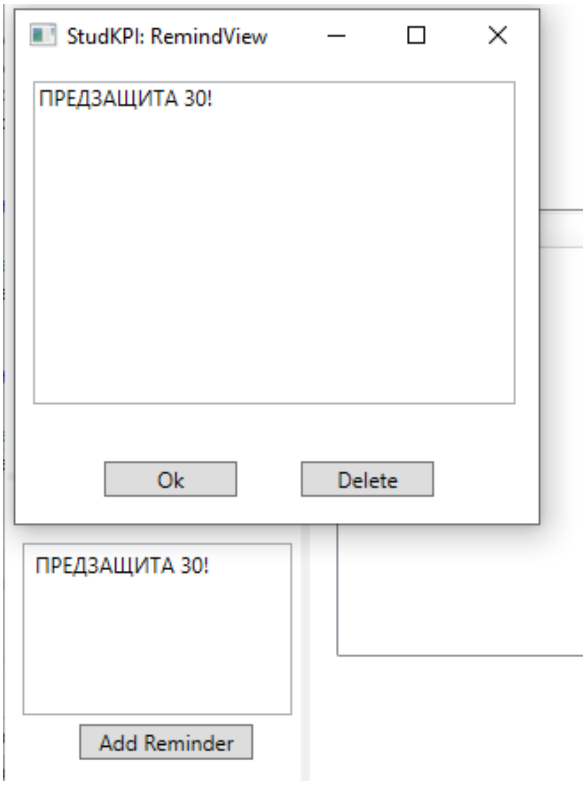


Рисунок 5.14 – Текст нагадування

При першому запуску програми, у вас ще немає заповненого розкладу занять, тому активна клавіша «Add Schedule» у вікні поточного курсу, при її натисканні відкриється вікно додавання розкладу, дивитись рисунок 5.15.

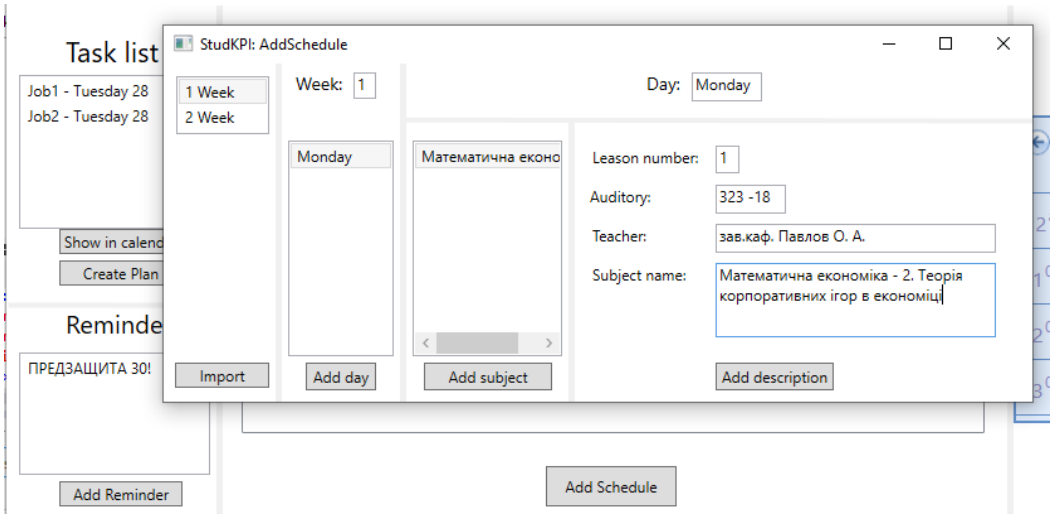


Рисунок 5.15 – Вікно додавання розкладу

Розклад можна водити власноруч або імпортувати з «api.rozklad.org.ua».

При імпорті розкладу всерівно можна буде редагувати інформацію. Після заповнення, розклад відображатиметься у вікні поточного курсу, дивитись рисунок 5.16.

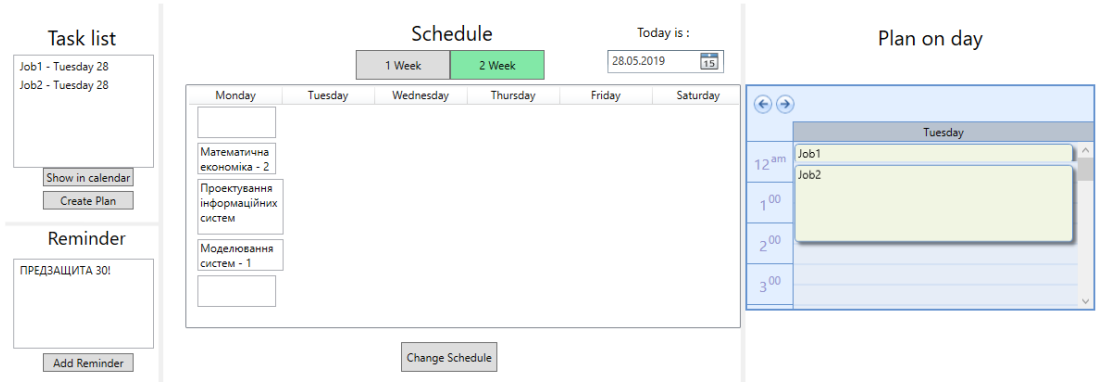


Рисунок 5.16 – Вікно поточного курсу з розкладом

Поточний номер тижня підсвічується зеленим. Клавіша «Add Schedule» змінилась на «Change Schedule», таким чином, ви можете доповнити розклад у інший час.

Натиснувши на предмет у розкладі, відкриється вікно відповідного предмета, дивитись рисунок 5.17.

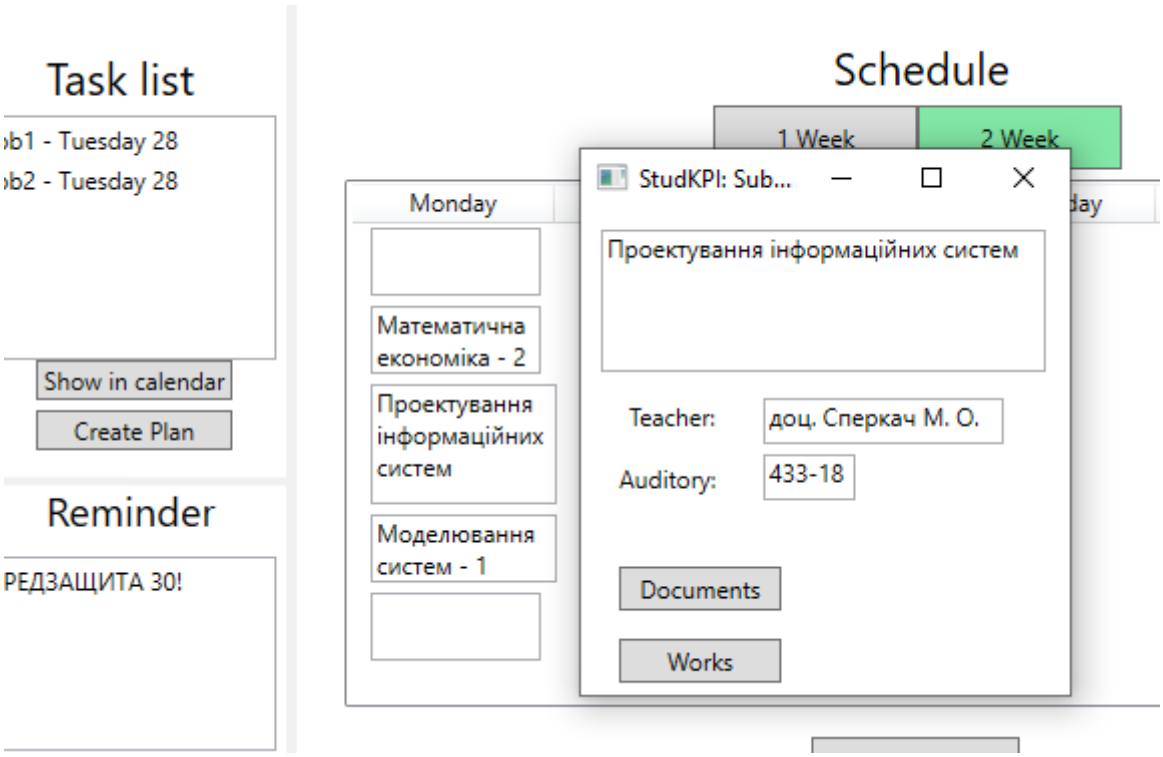


Рисунок 5.17 – Вікно обраної дисципліни

Натиснувши клавішу «Documents» можна додати та переглянути файли відповідної дисципліни, натиснувши на клавішу «Works» з'явиться вікно завдань, дивитись рисунок 5.18.

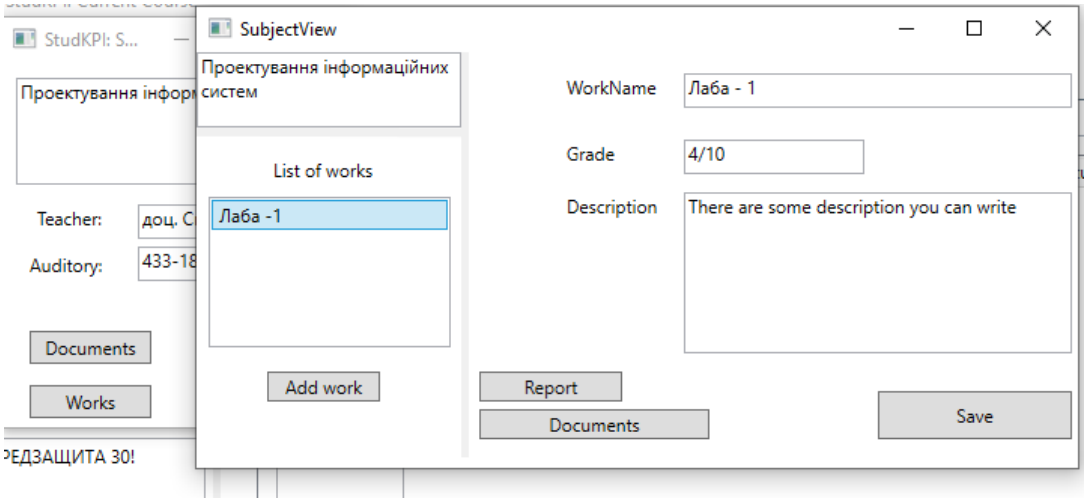


Рисунок 5.18 – Вікно завдань по дисципліні

Обравши завдання зі списку можна переглянути та змінити інформацію про нього, також можна переглянути та додати звіт та документи, натиснувши клавіші «Report» та «Documents» відповідно. Щоб додати нове завдання

натисніть «Add work», заповніть та додайте необхідну інформацію та натисність «Save».

Вся заповнена інформація зберігається, тому вона доступна у режимі перегляду у розділі «History» головного меню.

Переглянути інформацію по попереднім курсам, можна обравши відповідний курс зі списку, дивитись рисунок 5.19.

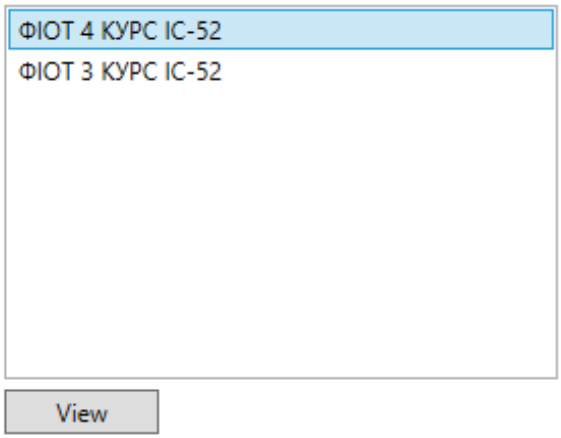


Рисунок 5.19 – Вікно перегляду історії курсів

Обравши курс, можна переглянути дисципліни курсу і так далі, дивитись рисунок 5.20.

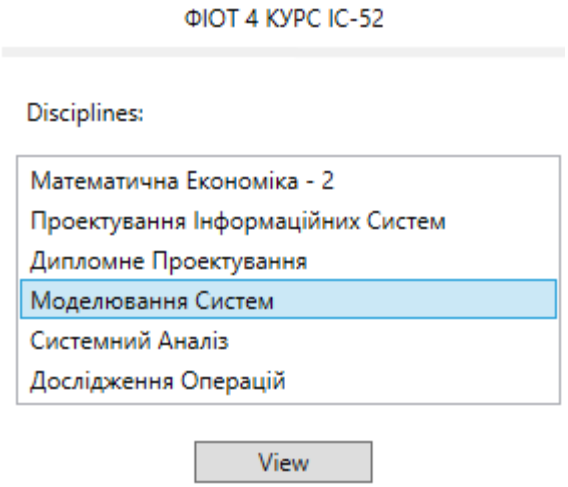


Рисунок 5.20 – Вікно перегляду дисциплін курсу

У розділі «Documents» головного меню можна переглядати та завантажити ієрарховані папки з файлами відповідних навчальних курсів, дивитись рисунок 5.21.

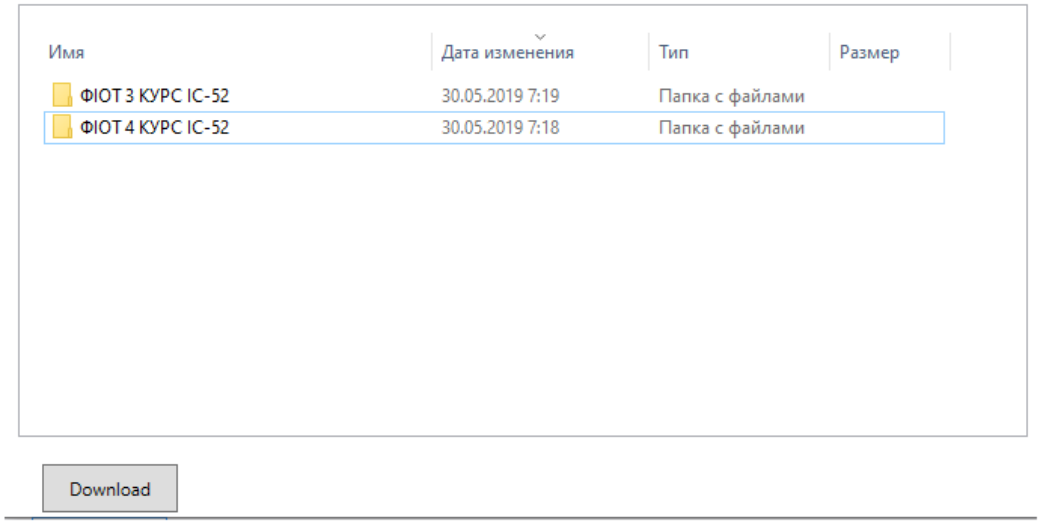


Рисунок 5.21 – Вікно перегляду папок курсів

Папки ієрарховані, дивитись рисунок 5.22.

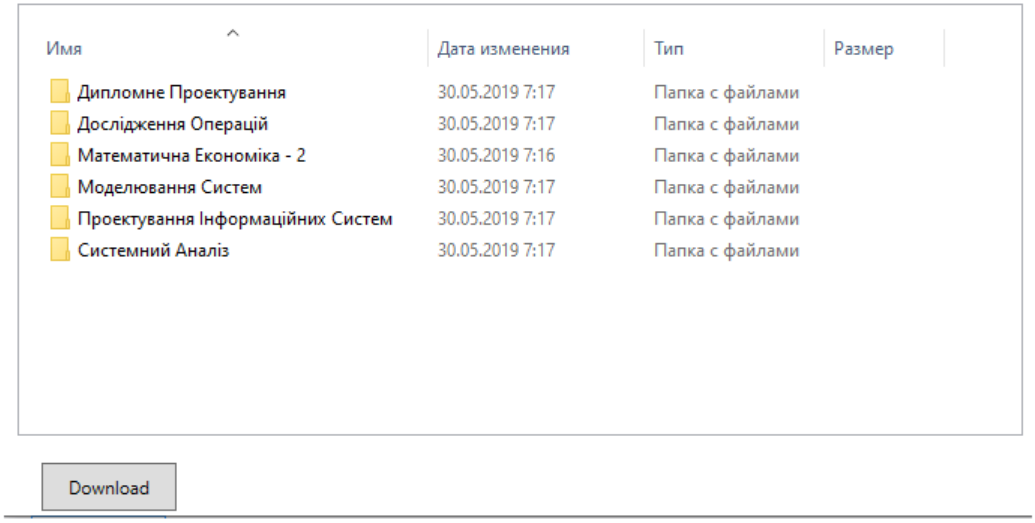


Рисунок 5.22 – Вікно перегляду папок дисциплін курсу

В кінці тижня, коли завершився план виконання робіт, можна переглянути графік навантаженості по дням цього плану, дивитись рисунок 5.23.

Рисунок 5.23 – Вікно перегляду графіку завантаженості днів роботами

Поточний графік можна порівняти з графіком минулого відповідного тижня, натиснувши клавішу «Compare to previous week», дивитись рисунок 5.24.

Рисунок 5.24 – Вікно порівняння графіків завантаженості днів роботами

Графіки відображають завантаженість за конкретний тиждень, порівнюють з попереднім. Завантаженість на день – число.

5.2 Випробування програмного продукту

5.2.1 Мета випробувань

Мета випробувань - перевірка функціональної відповідності інформаційної системи підтримки навчальної діяльності студента.

5.2.2 Загальні положення

Випробування проводяться на основі наступних документів:

– ГОСТ РД 50-34.698-90. Автоматизовані системи вимог до змісту документів;

– ГОСТ 34.603–92. Інформаційна технологія. Види випробувань автоматизованих систем.

5.2.3 Результати випробувань

Була протестована функціональна складова застосунку. У наступних таблицях приведені випробування деяких функціональних можливостей:

Таблиця 5.1 – Додавання нового завдання дисципліни

Мета тесту	Перевірка функції «Додавання завдання»
Початковий стан моделі	Відкрита сторінка обраної дисципліни
Вхідні дані:	Назва завдання, оцінка, коментар
Схема проведення тесту:	Ввести вхідні дані про нове завдання. Натиснути клавішу «Add»
Очікуваний результат:	Відображення нового завдання у списку завдань обраної дисципліни
Стан моделі після проведення випробувань:	Показане діалогове вікно про успішне додавання завдання, відкрита сторінка обраної дисципліни.

Таблиця 5.2 – Перевірка вибору навчальної дисципліни зі списку

Мета тесту	Перевірка функції «Вибір дисципліни»
Початковий стан моделі	Відкрита сторінка «Навчальні дисципліни».
Схема проведення тесту:	Натиснути на обрану дисципліну зі списку.
Очікуваний результат:	Перехід до сторінки обраної дисципліни.
Стан моделі після проведення випробувань:	Відображена сторінка обраної дисципліни.

Таблиця 5.3 – Перевірка імпорту навчального розкладу

Мета тесту	Перевірка функції «Імпорт навчального розкладу »
Початковий стан моделі	Відкрита головна сторінка застосунку. Таблиця розкладу порожня
Вхідні дані:	Назва навчальної групи
Схема проведення тесту:	Ввести назву групи у відповідне поле, натиснути клавшу «Import»
Очікуваний результат:	Заповнення таблиці розкладу
Стан моделі після проведення випробувань:	Головна сторінка застосунку має заповнену таблицю навчального розкладу

Таблиця 5.4 – Перегляд пройденого курсу

Мета тесту	Перевірка функції «Перегляд курсів»
Початковий стан моделі:	Відкрита початкова сторінка застосунку
Схема проведення тесту:	Натиснути клавішу «History»
Очікуваний результат:	Перехід до сторінки перегляду пройдених курсів

Продовження таблиці 5.4.

Стан моделі після проведення випробувань:	Відображення списку дисциплін курсу.

Таблиця 5.5 – Перевірка накопичених за діяльність файлів

Мета тесту:	Перевірка функції «Перегляд накопичених файлів»
Початковий стан моделі:	Відкрита головна сторінка застосунку
Схема проведення тесту:	Натиснути клавішу «Documents»
Очікуваний результат:	Відображення папок зі вмістом, що накопичений протягом процесу діяльності програми
Стан моделі після проведення випробувань:	Відображення папок з файлами

Висновок до розділу

В даному розділі описано керівництво користувача та привдено тестові варіанти з початковими умовами та результатами проходження тестів.

ЗАГАЛЬНІ ВИСНОВКИ

Даний дипломний проект розглядає роботу інформаційнї системи підтримки навчальної діяльності студента.

У розділі загальних положень розглядається основний процес діяльності користувача та функціональна модель за допомогою схеми варіантів використання. З її допомогою було показано функціональні вимоги до системи. Крім того був наведений короткий опис предметного середовища. Також було описано мету та призначення розробки. У розділі наявних аналогів були описані схожі за функціоналом застосування, які вирішують потрібну проблему. В кінці розділу була показана табличка з різними критеріями оцінювання схожих та розробленного застосувань.

У розділі інформаційного забезпечання були описані вхідні дані, вихідні дані, що надає система, та детально описано схему бази даних, що зберігаются в системі. Серед вихідних даних, крім накопичених протягом діяльності застосування вхідних даних, також є дані про плани виконання поставлених робіт. Також вихідними даними є графіки завантаженості роботами днів протягом тижня.

У математичному розділі наведено теоретичні відомості про пробели задач створення розкладу, та наведено їх математичний опис. Був описаний алгоритм розв'язку конкретної задачі та наведений приклад розв'язку поставленної задачі.

У розділі програмної та технічної підтримки були описані технології, що використовувались при розробці застосування, архітектури програмного забезпечення та побудована схема структурна класів для її відображення. Були представлені основні компоненти програмного забезпечення у вигляді схеми структурної компонентів. Основна взаємодія класів та об'єктів програмного забезпечення були відображені на схемі структурній послідовності. Представлені таблиці, що містять опис методів та їх параметрів, що представлені в схемі

структурній класів.

Програма написана на мові C# за допомогою платформи WPF. В якості сховища даних було обрано Microsoft SQL Server. Взаємодію з базою даних було реалізовано за допомогою технології Entity Framework. Також приведені вимоги до технічного забезпечення.

У технологічному розділі наведено вичерпне керівництво користувача та екранні форми застосунку. Основні тестові сценарії та результат їх проходження описаний у розділі випробування програмного продукту.

Після проходження всіх етапів розробки та тестування можна зазначити, що програмний продукт відповідає всім функціональним вимогам, які до нього ставили у технічному завданні.

ПЕРЕЛІК ПОСИЛАНЬ

1. Кисильова О. М. Задачі оптимального розбиття множин: теорія, алгоритми, застосування: монографія / О.М. Кисельова. —2005.
2. Лазарев А.А., Гафаров Е.Р., Теорія розкладів. Задачі і алгоритми, Москва, 2011.
3. Кочетов Ю.А. Методы локального поиска для дискретних задач размещения. Новосибирск, 2009. 261 стр.
4. Кочетов Ю.А. Вероятностные методы локального поиска для задач дискретной оптимизации.
5. Танаев В.С., Шкурба В.В., Введение в теорию расписаний, изд. «Наука», Москва, 1975.
6. Michael L. Pinedo. Scheduling: Theory, Algorithms, and Systems/ Michael L. Pinedo -2012.
7. Теория вероятностей и математическая статистика: Учебное пособие. Стандарт третьего поколения. — СПб.: Питер, 2013. — 192 с.
8. Руководство по WPF [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/wpf/
9. Паттерн MVVM [Електронний ресурс] – Режим доступу: https://metanit.com/sharp/wpf/22.1.php
10. Работа с Entity Framework 6 [Електронний ресурс] – Режим доступу: https://professorweb.ru/my/entity-framework/6/level1/
11. XAML in WPF [Електронний ресурс] – Режим доступу: https://docs.microsoft.com/en-us/dotnet/framework/wpf/advanced/xaml-in-wpf
12. JSON [Електронний ресурс] // Режим доступу: https://www.json.org/
13. Обзор LINQ to SQL[Електронний ресурс] // Режим доступу: https://professorweb.ru/my/LINQ/linq_sql/level9/9_1.php

14. Data Binding(WPF)[Електронний ресурс] // Режим доступу: https://docs.microsoft.com/en-us/dotnet/framework/wpf/data/data-binding-wpf

15. Теорія розкладів: Конспект лекцій / О. Г. Жданова. – ФІОТ АСОІУ НТУУ «Київський політехнічний інститут імені Ігоря Сікорського», 2018.

Додаток А

Тексти програмного коду

Інформаційна система підтримки навчальної діяльності студента

(Найменування програми (документа))

DVD-R

(Вид носія даних)

10 аркушів, 2 мб

(Обсяг програми (документа) , арк.,) Кб)

Київ – 2019 року

Calendar.cs

```
public class Calendar : Control
   {
      static Calendar()
      {
         DefaultStyleKeyProperty.OverrideMetadata(typeof(Calendar), new Frame-
workPropertyMetadata(typeof(Calendar)));

         CommandManager.RegisterClassCommandBinding(typeof(Calendar), new
CommandBinding(NextDay, new ExecutedRoutedEventHandler(OnExecut-
edNextDay), new CanExecuteRoutedEventHandler(OnCanExecuteNextDay)));
         CommandManager.RegisterClassCommandBinding(typeof(Calendar), new
CommandBinding(PreviousDay, new ExecutedRoutedEventHandler(OnExecut-
edPreviousDay), new CanExecuteRoutedEventHandler(OnCanExecutePrevi-
ousDay)));
      }

      #region AddAppointment

      public static readonly RoutedEvent AddAppointmentEvent =
         CalendarTimeslotItem.AddAppointmentEvent.AddOwner(typeof(Cal-
endarDay));

      public event RoutedEventHandler AddAppointment
      {
         add
         {
            AddHandler(AddAppointmentEvent, value);
         }
         remove
         {
            RemoveHandler(AddAppointmentEvent, value);
         }
      }

      #endregion

      #region Appointments

      public static readonly DependencyProperty AppointmentsProperty =
         DependencyProperty.Register("Appointments", typeof(IEnumerable<Ap-
pointment>), typeof(Calendar),
```

```
            new FrameworkPropertyMetadata(null, new Property-
ChangedCallback(Calendar.OnAppointmentsChanged)));

        public IEnumerable<Appointment> Appointments
        {
            get { return (IEnumerable<Appointment>)GetValue(AppointmentsProp-
erty); }
            set { SetValue(AppointmentsProperty, value); }
        }

        private static void OnAppointmentsChanged(DependencyObject d, Depend-
encyPropertyChangedEventArgs e)
        {
            ((Calendar)d).OnAppointmentsChanged(e);
        }

        protected virtual void OnAppointmentsChanged(DependencyProperty-
ChangedEventArgs e)
        {
            FilterAppointments();
        }

        #endregion

        #region CurrentDate

        /// <summary>
        /// CurrentDate Dependency Property
        /// </summary>
        public static readonly DependencyProperty CurrentDateProperty =
            DependencyProperty.Register("CurrentDate", typeof(DateTime),
typeof(Calendar),
                new FrameworkPropertyMetadata((DateTime)DateTime.Now,
                    new PropertyChangedCallback(OnCurrentDateChanged)));

        /// <summary>
        /// Gets or sets the CurrentDate property.  This dependency property
        /// indicates ....
        /// </summary>
        public DateTime CurrentDate
        {
            get { return (DateTime)GetValue(CurrentDateProperty); }
            set { SetValue(CurrentDateProperty, value); }
```

```
 }
        /// <summary>
        /// Handles changes to the CurrentDate property.
        /// </summary>
        private static void OnCurrentDateChanged(DependencyObject d, Dependen-
cyPropertyChangedEventArgs e)
        {
           ((Calendar)d).OnCurrentDateChanged(e);
        }

        /// <summary>
        /// Provides derived classes an opportunity to handle changes to the Current-
Date property.
        /// </summary>
        protected virtual void OnCurrentDateChanged(DependencyProperty-
ChangedEventArgs e)
        {
           FilterAppointments();
        }

        #endregion

        private void FilterAppointments()
        {
           DateTime byDate = CurrentDate;
           CalendarDay day = this.GetTemplateChild("day") as CalendarDay;
           day.ItemsSource = Appointments.ByDate(byDate);

           TextBlock dayHeader = this.GetTemplateChild("dayHeader") as Text-
Block;
           dayHeader.Text = byDate.DayOfWeek.ToString();
        }

        #region NextDay/PreviousDay

        public static readonly RoutedCommand NextDay = new RoutedCom-
mand("NextDay", typeof(Calendar));
        public static readonly RoutedCommand PreviousDay = new RoutedCom-
mand("PreviousDay", typeof(Calendar));

        private static void OnCanExecuteNextDay(object sender, CanExecut-
eRoutedEventArgs e)
        {
```

```
        ((Calendar)sender).OnCanExecuteNextDay(e);
    }

    private static void OnExecutedNextDay(object sender, ExecutedRoutedEven-
tArgs e)
    {
        ((Calendar)sender).OnExecutedNextDay(e);
    }

    protected virtual void OnCanExecuteNextDay(CanExecuteRoutedEventArgs
e)
    {
        e.CanExecute = true;
        e.Handled = false;
    }

    protected virtual void OnExecutedNextDay(ExecutedRoutedEventArgs e)
    {
        CurrentDate += TimeSpan.FromDays(1);
        e.Handled = true;
    }

    private static void OnCanExecutePreviousDay(object sender, CanExecut-
eRoutedEventArgs e)
    {
        ((Calendar)sender).OnCanExecutePreviousDay(e);
    }

    private static void OnExecutedPreviousDay(object sender, Execut-
edRoutedEventArgs e)
    {
        ((Calendar)sender).OnExecutedPreviousDay(e);
    }

    protected virtual void OnCanExecutePreviousDay(CanExecuteRoutedEven-
tArgs e)
    {
        e.CanExecute = true;
        e.Handled = false;
    }

    protected virtual void OnExecutedPreviousDay(ExecutedRoutedEventArgs e)
    {
```

```
            CurrentDate -= TimeSpan.FromDays(1);
            e.Handled = true;
        }

        #endregion
```

BindableObject.cs

```
[Serializable]
    public abstract class BindableObject : INotifyPropertyChanged
    {
        #region Data

        private static readonly Dictionary<string, PropertyChangedEventArgs> even-
tArgCache;
        private const string ERROR_MSG = "{0} is not a public property of {1}";
        private static readonly object syncLock = new object();

        #endregion // Data

        #region Constructors

        static BindableObject()
        {
            eventArgCache = new Dictionary<string, PropertyChangedEventArgs>();
        }

        protected BindableObject()
        {
        }

        #endregion // Constructors

        #region Public Members

        [field: NonSerialized]
        public event PropertyChangedEventHandler PropertyChanged;

        public static PropertyChangedEventArgs GetPropertyChangedEven-
tArgs(string propertyName)
        {
            if (String.IsNullOrEmpty(propertyName))
            {
```

```
            throw new ArgumentException("propertyName cannot be null or
empty.");
         }

         PropertyChangedEventArgs args;
         lock (BindableObject.syncLock)

         {
            if (!eventArgCache.TryGetValue(propertyName, out args))
            {
               eventArgCache.Add(propertyName, args = new Property-
ChangedEventArgs(propertyName));
            }
         }

         return args;
      }

      #endregion // Public Members

      #endregion // Private Helpers
   }
```

SetDivisionTaskInputs.cs

```
public class SetDivisionTaskInputs
   {
      public int NumberOfSubDivision { get; set; }

      private IEnumerable<int> _elementsSet;
      public IEnumerable<int> ElementsSet
      {
         get { return _elementsSet; }
         set
         {
            NumberOfElements = (value != null) ? value.Count() : 0;
            _elementsSet = value;
         }
      }

      public int NumberOfIterations { get; set; }

      public int NumberOfElements { get; private set; }
```

```
        public SetDivisionTaskInputs() : this(0, null)
        {
        }

        public SetDivisionTaskInputs(int numberOfSubDivision, IEnumerable<int>
elementsSet)
        {
            NumberOfSubDivision = numberOfSubDivision;
            ElementsSet = elementsSet ?? new int[] { };
        }
    }
```

SetDivisionTaskSolver.cs

```
public class SetDivisionTaskSolver
    {

        private SetDividerTaskSolution firstSolution;
        private SetDividerTaskSolution currentSolution;
        private SetDividerTaskSolution bestCurrentSolution;

        public SetDivisionTaskInputs Inputs { get; private set; }
        public SetDivisionTaskOutputs Outputs { get; private set; }

        private List<SetDividerTaskSolution> solutionList;
        private int currentIteration = 1;

        private List<SetDividerTaskSolution> tempSolutions;
        private HashSet<SetDividerTaskSolution> tabuList;

        public SetDivisionTaskSolver()
        {
            solutionList = new List<SetDividerTaskSolution>();
            tempSolutions = new List<SetDividerTaskSolution>();
        }

        public SetDivisionTaskOutputs Solve(SetDivisionTaskInputs input, Algo-
rithm algorithm)
        {
            // Set up variables
            Inputs = input;

            currentSolution = new SetDividerTaskSolution(new bool[Inputs.Number-
OfElements, Inputs.NumberOfSubDivision],
```

```
            new int[Inputs.NumberOfSubDivision],
            input);
            List = new HashSet<SetDividerTaskSolution>(new BelongingMa-
trixComparer());


        // First Stage
        LazySetUp();
        bestCurrentSolution = currentSolution;

        // Second Stage
        do
        {
            if (numberOfBestSolution == null)
                break;

            if (currentSolution.GoalFuncionValue > tempSolutions[numberOf-
BestSolution.Value].GoalFuncionValue)
            {
                if (bestCurrentSolution.GoalFuncionValue > tempSolutions[number-
OfBestSolution.Value].GoalFuncionValue)
                {
                    bestCurrentSolution = tempSolutions[numberOfBestSolu-
tion.Value];
                    currentIteration = 1;
                }
            }

            currentSolution = tempSolutions[numberOfBestSolution.Value];
            solutionList.Add(tempSolutions[numberOfBestSolution.Value]);

            List.Add(currentSolution);
        } while (true);

            currentSolution = bestCurrentSolution;

            solutionList.Add(currentSolution);

        Outputs = new SetDivisionTaskOutputs(firstSolution, currentSolution, so-
lutionList);

        return Outputs;
    }
```

```
        private void DoExchanges(bool checkTabu)
        {
            tempSolutions.Clear();

            int smallestSubSet = NumberOfSmallesSubSet(currentSolution);

            int biggestSubSet = NumberOfBiggestSubSet(currentSolution);

            float accuracy = Inputs.AccuracyLevel / 100f;
            if (accuracy < 0.1f)
                accuracy = 0.1f;
            else if (accuracy > 1f)
                accuracy = 1f;

            List<int> indexes = new List<int>();
            Random r = new Random();

            indexes.AddRange(Enumerable.Range(0, Inputs.NumberOfElements - 1)
                        .OrderBy(i => r.Next())
                        .Take((int)Math.Ceiling(Inputs.NumberOfElements * accu-
racy)));

            // All Exchanges
            for (int i = 0; i < indexes.Count; i++)
            {
                for (int j = 0; j < indexes.Count; j++)
                {
                    if (currentSolution.BelongingMatrix[i, smallestSubSet] && currentSo-
lution.BelongingMatrix[j, biggestSubSet])
                    {
                        TryExchange(i, smallestSubSet, j, biggestSubSet, currentSolution,
checkTabu);
                    }
                }
            }

            // All Donates
            for (int i = 0; i < indexes.Count; i++)
            {
                if (currentSolution.BelongingMatrix[i, biggestSubSet])

                {
```

```
                TryDonate(i, biggestSubSet, smallestSubSet, currentSolution, checkTabu);

            }
        }
    }

    private int? SelectBestSolution(bool isTabu)
    {
        int? number = null;
        SetDividerTaskSolution maxOfAll = tempSolutions.Max();

            currentIteration++;

        for (int i = 0; i < tempSolutions.Count; i++)
        {
            if (maxOfAll == tempSolutions[i])
                number = i;
        }

        return number;
    }

    private void LazySetUp()
    {
        IEnumerable<int> set = Inputs.ElementsSet;

        for (int i = 0; i < Inputs.NumberOfElements; i++)
        {
            int element = set.ElementAt(i);
            int smallestSet = NumberOfSmallesSubSet(currentSolution);

            currentSolution.BelongingMatrix[i, smallestSet] = true;
            currentSolution.WeightList[smallestSet] += element;
        }

        firstSolution = (SetDividerTaskSolution)currentSolution.Clone();
        if (tabuList != null)
            tabuList.Add(firstSolution);
    }
}
```

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО
Керівник проекту
_____ *О.М. Клименко*
(підпис) (ініціали, прізвище)
“16” квітня 2019 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А. Павлов*
(підпис) (ініціали, прізвище)
“17” квітня 2019 р.

Інформаційна система підтримки навчальної діяльності студента

ТЕХНІЧНЕ ЗАВДАННЯ

Шифр *ДП ІС-5202.1181-с.ТЗ*

на 10 сторінках

Київ – 2019 року

ЗМІСТ

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ

1.1 Повне найменування системи та її умовне позначення

Повна назва системи: Інформаційна система підтримки навчальної діяльності студента

1.2 Найменування організації-замовника та організації-учасника робіт

Генеральним замовником проекту є кафедра Автоматизованих систем обробки інформації та управління НТУУ «КПІ ім. Ігоря Сікорського». Представником замовника є Клименко Олена Миколаївна.

Розробником системи є студент факультету інформатики та обчислювальної техніки НТУУ «КПІ ім. Ігоря Сікорського»: студент групи ІС-52 Бобер Євгеній Олександрович.

1.3 Перелік документів, на підставі яких створюється система

При розробці системи і створення проектно-експлуатаційної документації Виконавець повинен керуватися вимогами наступних нормативних документів:

- ДСТУ 19.201-78. Технічне завдання. Вимоги до змісту і оформлення;
- ДСТУ 34.601-90. Комплекс стандартів на автоматизовані системи. Автоматизовані системи. Стадії створення;
- ДСТУ 34.201-89. Інформаційні технології. Комплекс стандартів на автоматизовані системи. Види, комплексність і позначення документів при створенні автоматизованих систем.

1.4 Планові терміни початку і закінчення роботи зі створення системи

Плановий термін початку роботи над інформаційною системою підтримки навчальної діяльності студента – 5 лютого 2019 рік.

Плановий термін по закінченню роботи над інформаційною системою підтримки навчальної діяльності студента – не пізніше 1 червня 2019 року.

2 ПРИЗНАЧЕННЯ І МЕТА СТВОРЕННЯ КОМПЛЕКСУ ЗАДАЧ

2.1 Призначення комплексу задач

Призначенням системи є забезпечення контролю навчальної діяльності студентів та покращення процесу планування навчання студентів.

2.2 Цілі створення комплексу задач

Цілями розробки є:

- накопити інформацію про навчальну діяльність студентів;
- покращити організацію виконання навчальної роботи студентів;
- спростити процес пошуку навчальної інформації та інформації з діяльності студентів.

Для досягнення поставлених цілей необхідно розв’язати наступні задачі:

- введення навчального розкладу;
- введення інформації за навчальними дисциплінами та додавання завдань;
- створення плану виконання поставлених робіт;
- формування архіву поточних та попередніх завершених навчальних дисциплін.

3 ХАРАКТЕРИСТИКА ОБ'ЄКТА АВТОМАТИЗАЦІЇ

Для користування системою обов'язковою умовою для користувача є наявність операційної системи Windows(7 та вище).

Працювати з системою можна користувачам, що встановили даний продукт. Після встановлення системи, користувач має доступ до функціоналу системи.

Об'єктом автоматизації є процес накопичення інформації про діяльність студента та створення плану виконання робіт студентом.

4 ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Вимоги до функціональних характеристик

Система має виконувати наступні функції:

- отримання первинних даних;
- накопичення інформації про навчальну діяльності студентів:
    1) накопичення інформації по семестрах та дисциплінах;
    2) накопичення файлових матеріалів;
- створення плану виконання робіт;
- перегляд та експорт накопичених файлів.

4.2 Вимоги до надійності

Програма повинна зберігати працездатність і забезпечувати відновлення своїх функцій при виникненні наступних позаштатних ситуацій:

– при помилках в роботі апаратних засобів (крім носіїв даних і програм).

Продукт повинен поєднувати надійність і функціональність. У разі виникнення аварійних ситуацій необхідно сповіщати користувача та надавати інструкцію для подальших дій. Будь-які аварійні ситуації мають бути задокументовані у звіті, який при необхідності надсилається розробнику для визначення причини збою в роботі та усуненні помилок, які могли привести до нестабільної роботи програмного продукту.

4.3 Вимоги до складу і параметрів технічних засобів

Для правильної роботи системи до складу технічних засобів повинен входити комп’ютер, що має конфігурацію наведену нижче:

- комп’ютер з такою конфігурацією:
    1) процесор з тактовою частотою не нижче 2 ГГц;
    2) об’єм оперативної пам’яті не менше 512 Мб;

3) інші складові можуть мати будь-які параметри, тому що вони не значним чином впливають на роботу застосування;

- комп’ютерна периферія, до складу якої входить:
  1) монітор;
  2) мишка;
  3) клавіатура.

5 СТАДІЇ І ЕТАПИ РОЗРОБКИ

Основні етапи виконання робіт з інформаційної системи підтримки навчальної діяльності студента.

Таблиця 5.1 – Стадії та етапи розробки

Та№ п/п	**Назва етапу роботи**	**Термін виконання етапу**	**Результат виконання**
1.	Підготовка технічного завдання на розробку програмного продукту	07.02.2019	
2.	Розробка сценарію роботи	12.02.2019	
3.	Технічне проектування – функціональність, модулі, задачі, цілі тощо	20.02.2019	
4.	Узгодження з керівником інтерфейсу користувача	02.03.2019	
5.	Розробка інформаційного забезпечення	10.03.2019	
6.	Розробка програмного забезпечення	10.04.2019	
7.	Налагодження програми	20.04.2019	
8.	Тестування програми	16.05.2019	
9.	Здача готового програмного продукту замовнику	30.05.2019	

5.1 ПОРЯДОК КОНТРОЛЮ ТА ПРИЙМАННЯ СИСТЕМИ

5.2 Види випробувань

Види, склад, об’єм і методи випробувань програмного продукту повинні бути викладені в програмі і методиці випробувань системи, що розробляється в складі робочої документації.

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ СІКОРСЬКОГО”
Кафедра автоматизованих систем обробки інформації та управління

УЗГОДЖЕНО
Керівник проекту
_____ *О.М. Клименко*
(підпис) (ініціали, прізвище)
“13” травня 2019 р.

ЗАТВЕРДЖУЮ
В.о. завідувача кафедри
_____ *О.А.Павлов*
(підпис) (ініціали, прізвище)
“14” травня 2019 р.

Інформаційна система підтримки навчальної діяльності студента

ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

Шифр *ДП ІС-5202.1181-с.ПМВ*

на 10 сторінках

Київ – 2019 року

ЗМІСТ

1 ОБ'ЄКТ ВИПРОБУВАННЯ

1.1 Найменування програми

Інформаційна система підтримки навчальної діяльності студента.

1.2 Область застосування

Функціонал системи призначений для систематизації та полегшення ходу навчальної діяльності студента. Студент може накопичувати потрібну йому інформацію, створюючи при цьому свій особистий інформаційний ресурс. Може переглядати розклад навчання та користуватися планувальником поставлених робіт. Також користувач може вивантажити накопичені документи та файли та використовувати їх у своїх цілях.

1.3 Умовне позначення програми

Умовне позначення: «StudHouse».

2 МЕТА ВИПРОБУВАНЬ

Метою випробувань є перевірка відповідності функцій інформаційної системи підтримки навчальної діяльності студента вимогам технічного завдання.

3 ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

3.1 Вимоги до функціональних характеристик

Функціональні вимоги:

- можливість введення навчального розкладу;
- можливість перегляду навчального розкладу;
- можливість введення інформації за навчальними дисциплінами та додавання завдань;
- перегляд інформації за навчальними дисциплінами та завданнями;
- можливість створення плану виконання поставлених робіт;
- перегляд плану виконання поставлених робіт;
- можливість вивантаження та перегляд архіву поточних та попередніх завершених навчальних дисциплін;
- формування архіву поточних та попередніх завершених навчальних дисциплін.

3.1.1 Вимоги до складу виконуваних функцій

Функціонал повинен забезпечувати можливість виконання перерахованих нижче функцій:

- введення навчального розкладу;
- введення інформації за навчальними дисциплінами та додавання завдань;
- створення плану виконання поставлених робіт;
- вивантаження та перегляд архіву поточних та попередніх завершених навчальних дисциплін.

4 ВИМОГИ ДО ПРОГРАМНОЇ ДОКУМЕНТАЦІЇ

Програмна документація має складатися з керівництва користувача та вихідних текстів програмного коду.

5 СКЛАД І ПОРЯДОК ВИПРОБУВАНЬ

Етапи випробувань:

– ознайомчий;

– виконавчий.

На ознайомчому етапі проводиться:

– перевірка комплектності програмної документації;

– перевірка комплектності складу технічних і програмних засобів.

Під час виконавчого етапу проводиться:

– перевірка відповідності технічних характеристик системи;

– перевірка ступеню виконання вимог функціонального призначення системи.

Функції, що підлягають перевірці:

- введення навчального розкладу;
- введення інформації за навчальними дисциплінами та додавання завдань;
- створення плану виконання поставлених робіт;
- вивантаження та перегляд архіву поточних та попередніх завершених навчальних дисциплін.

6 МЕТОДИ ВИПРОБУВАНЬ

У наступних таблицях приведені випробування деяких функціональних можливостей:

Таблиця 5.1 – Додавання нового викладача

Мета тесту	Перевірка функції «Додавання завдання»
Початковий стан моделі	Відкрита сторінка обраної дисципліни
Вхідні дані:	Назва завдання, оцінка, коментар
Схема проведення тесту:	Ввести вхідні дані про нове завдання. Натиснути клавішу «Add»
Очікуваний результат:	Відображення нового завдання у списку завдань обраної дисципліни
Стан моделі після проведення випробувань:	Показане діалогове вікно про успішне додавання завдання, відкрита сторінка обраної дисципліни.

Таблиця 5.2 – Перевірка вибору навчальної дисципліни зі списку

Мета тесту	Перевірка функції «Вибір дисципліни»
Початковий стан моделі	Відкрита сторінка «Навчальні дисципліни».
Схема проведення тесту:	Натиснути на обрану дисципліну зі списку.
Очікуваний результат:	Перехід до сторінки обраної дисципліни.

Таблиця 5.3 – Перевірка імпорту навчального розкладу

Мета тесту	Перевірка функції «Імпорт навчального розкладу »
Початковий стан моделі	Відкрита головна сторінка застосунку. Таблиця розкладу порожня
Вхідні дані:	Назва навчальної групи
Схема проведення тесту:	Ввести назву групи у відповідне поле, натиснути клавшу «Import»
Очікуваний результат:	Заповнення таблиці розкладу
Стан моделі після проведення випробувань:	Головна сторінка застосунку має заповнену таблицю навчального розкладу

Таблиця 5.4 – Перегляд пройденого курсу

Мета тесту	Перевірка функції «Перегляд курсів»
Початковий стан моделі:	Відкрита початкова сторінка застосунку
Схема проведення тесту:	Натиснути клавішу «History»
Очікуваний результат:	Перехід до сторінки перегляду пройдених курсів
Стан моделі після проведення випробувань:	Відображення списку дисциплін курсу.

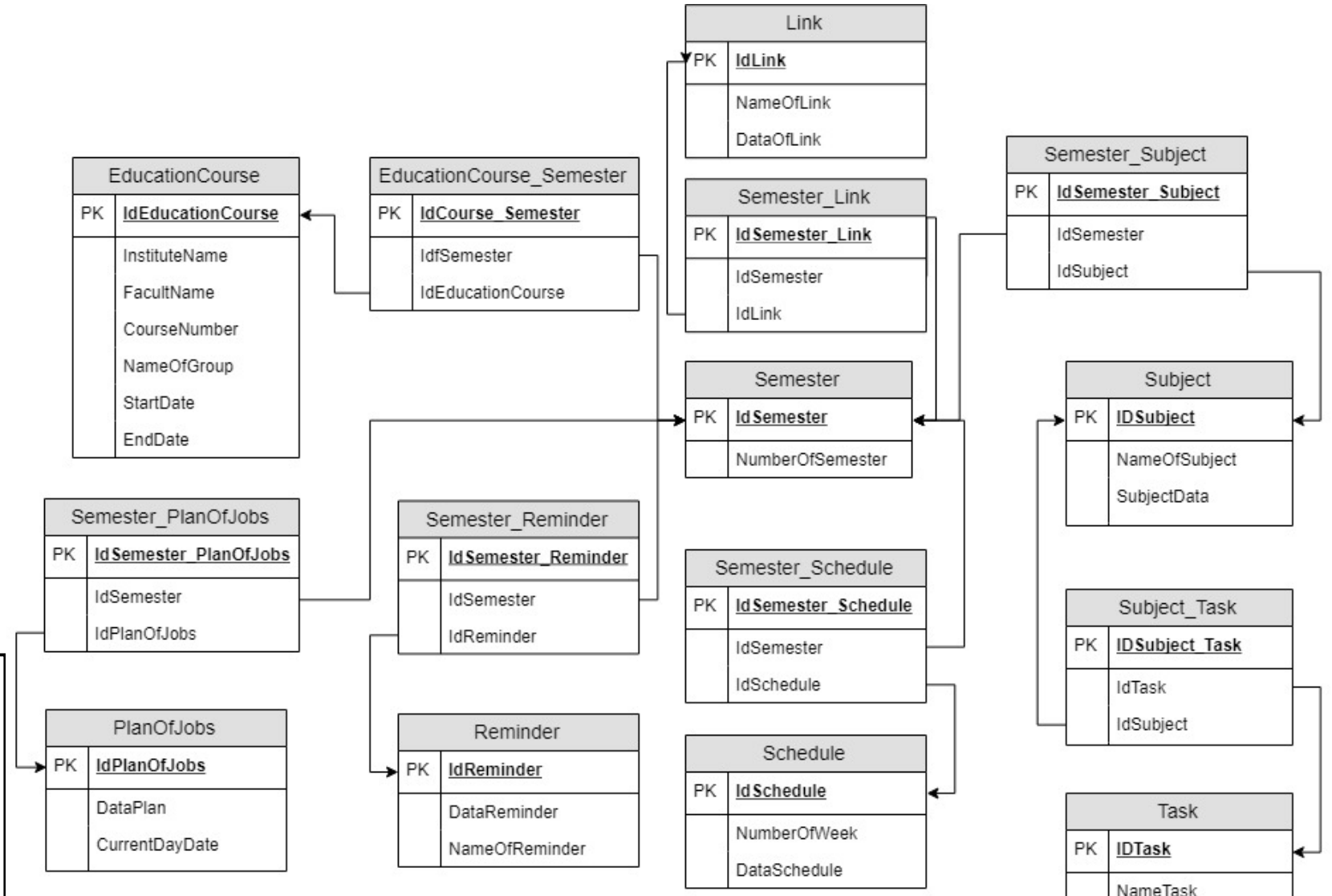
Таблиця 5.5 – Перевірка накопичених за діяльність файлів

Мета тесту:	Перевірка функції «Перегляд накопичених файлів»
Початковий стан моделі:	Відкрита головна сторінка застосунку
Схема проведення тесту:	Натиснути клавішу «Documents»
Очікуваний результат:	Відображення папок зі вмістом, що накопичений протягом процесу діяльності програми
Стан моделі після проведення випробувань:	Відображення папок з файлами

Графічний матеріал
до дипломного проекту

на тему: Інформаційна система підтримки навчальної діяльності студента

Київ – 2019 року



ДП ІС-5202.1181-с.СБД

ДП ІС-5202.1181-с.СБД

Схема бази даних

Інформаційна система підтримки навчальної діяльності студента

КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52

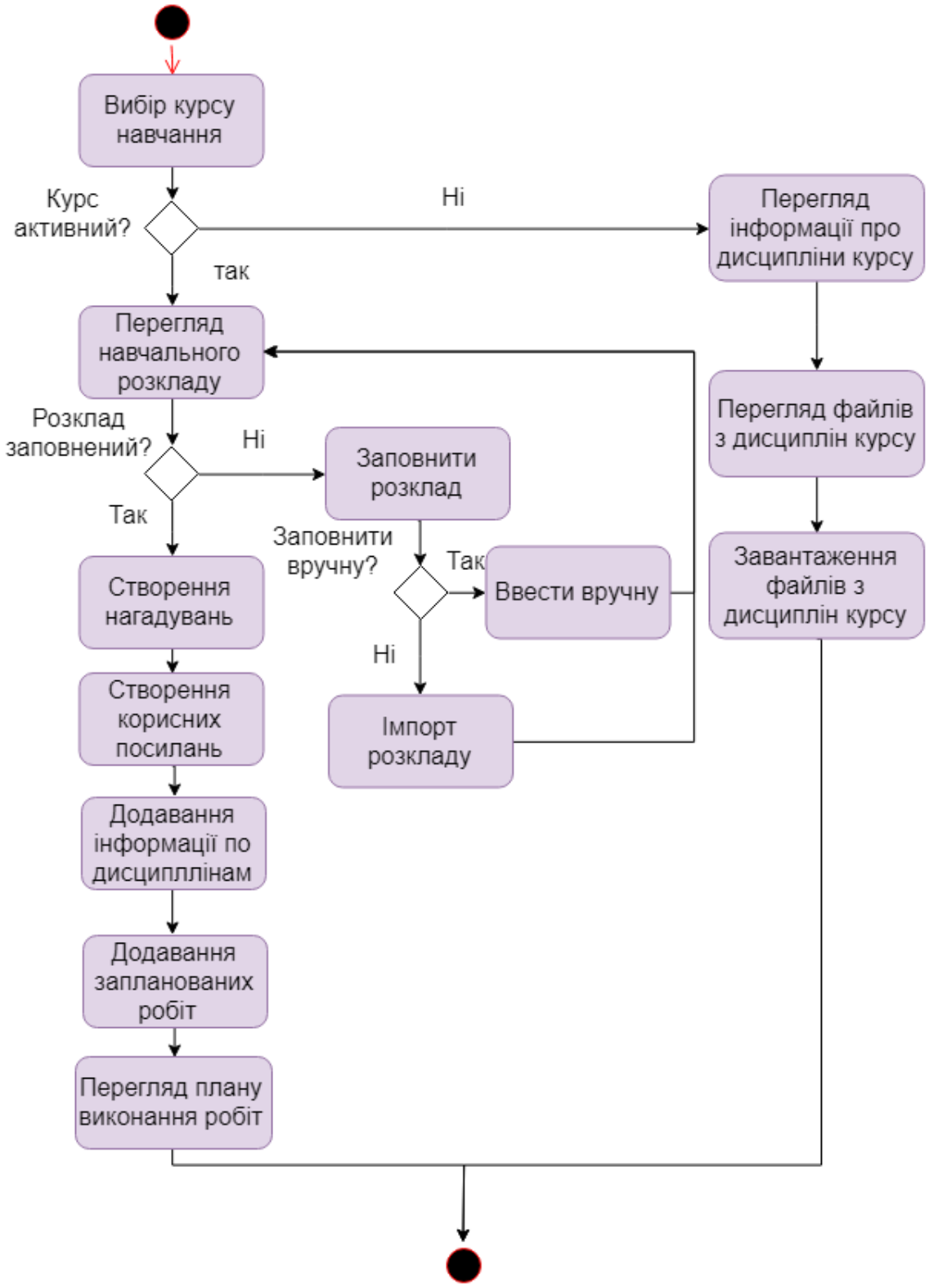
Літера	Маса	Масштаб

Аркуш 1 | Аркушів 1

Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Бобер Є.О.		
Перевірив		Клименко О.М.		
Т. кон.				
Н. кон.		Халус О.А.		
Затвердив		Клименко О.М.		

ДП ІС-5202.1181-с.ССВ

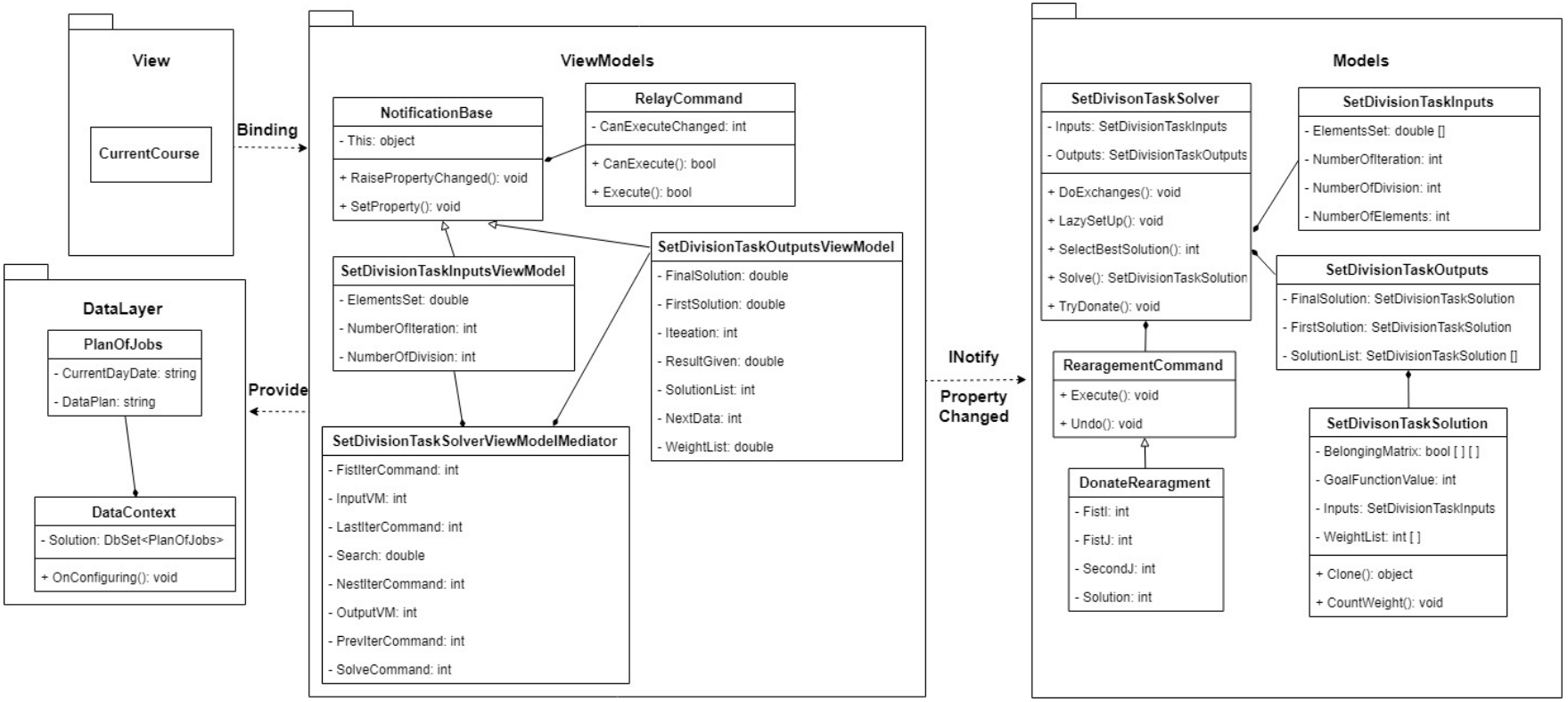
					ДП ІС-5202.1181-с.ССВ
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна варіантів використання
Розробив		Бобер Є.О.			Літера: — Маса: — Масштаб: —
Перевірив		Клименко О.М.			Аркуш 1 / Аркушів 1
Т. кон.					Інформаційна система підтримки навчальної діяльності студента
Н. кон.		Халус О.А.			КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Затвердив		Клименко О.М.			



					ДП ІС-5202.1181-с.ССД	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна діяльності	Літера / Маса / Масштаб
Розробив		Бобер Є.О.				
Перевірив		Клименко О.М.				Аркуш 1 / Аркушів 1
Т. кон.					Інформаційна система підтримки навчальної діяльності студента	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. кон.		Халус О.А.				
Затвердив		Клименко О.М.				

ДП ІС-5202.1181-с.ССП

					ДП ІС-5202.1181-с.ССП	
Зм.	Арк.	№ документа	Підпис	Дата		
Розробив		Бобер Є.О.			Схема структурна послідовності	Літера / Маса / Масштаб
						Аркуш 1 / Аркушів 1
Перевірив		Клименко О.М.				
Т. кон.					Інформаційна система підтримки навчальної діяльності студента	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. кон.		Халус О.А.				
Затвердив		Клименко О.М.				



ДП ІС-5202.1181-с.ССК

ДП ІС-5202.1181-с.ССК

Схема структурна класів

КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52

Інформаційна система підтримки навчальної діяльності студента

Зм.	Арк.	№ документа	Підпис	Дата
Розробив		Бобер Є.О.		
Перевірив		Клименко О.М.		
Т. кон.				
Н. кон.		Халус О.А.		
Затвердив		Клименко О.М.		

Літера	Маса	Масштаб
Аркуш 1		Аркушів 1

ДП ІС-5202.1181-с.ССК

					ДП ІС-5202.1181-с.ССК	
Зм.	Арк.	№ документа	Підпис	Дата	Схема структурна компонентів	Літера / Маса / Масштаб
Розробив		Бобер Є.О.				
Перевірив		Клименко О.М.				Аркуш 1 / Аркушів 1
Т. кон.					Інформаційна система підтримки навчальної діяльності студента	КПІ ім. Ігоря Сікорського кафедра АСОІУ гр. ІС-52
Н. кон.		Халус О.А.				
Затвердив		Клименко О.М.				